David Lawrence

Commodore Sachbuchreihe Band 11

DER COMMODORE 64 IN DER PRAXIS

Wertvolle Ideen und Anwendungen anwendbar auch auf dem Commodore 128

David Lawrence

DER COMMODORE 64 IN DER PRAXIS

Commodore Sachbuchreihe Band 11

David Lawrence

DER COMMODORE 64 IN DER PRAXIS

Wertvolle Ideen und Anwendungen anwendbar auch auf dem Commodore 128



Titel der Originalausgabe: The Working Commodore 64
Copyright c) David Lawrence, 1983
First published 1984 by:
Sunshine Books (an imprint of Scot Press Ltd.)
12–13 Little Newport Street, London WC2R 3LD

Aus dem Englischen übertragen von A. H. Kayser und Koll.

Artikel-Nr. 556423/4.85 Änderungen vorbehalten ISBN-Nr. 3-89133-011-1

INHALT

PROGRAMMHINWEISE

Bei dem Commodore 64, wie bei anderen Commodore-Systemen, unterliegt eine Reihe von Funktionen 'Steuerzeichen', die in normalen Zeichenfolgen enthalten sind und beim Ausdrucken der Zeichenfolge wirksam werden. Steuerzeichen werden normalerweise dadurch erkannt, daß sie invers dargestellt werden (die Hintergrund- und Vordergrundfarben sind bei der Zeichenposition umgekehrt). Die Funktionen, die von derartigen Zeichen gesteuert werden, umfassen die Cursorposition, die Druckfarbe, die Ein- und Ausschaltung der inversen DArstellung (RVS), die Rückführung des Cursors in die Ausgangsposition und das Löschen des Bildschirms.

BEZEICHNUNG	SYMBOL		TASTEN	ODER CHR$(XXX)
[SCHIRM NEU]		=	<SHIFT>+<CLR>	CHR$(147)
[HOME]		=	<CLR>	CHR$(19)
[CRSR HOCH]		=	<SHIFT>+<CRSR>	CHR$(145)
[CRSR RUNTER]		=	<CRSR>	CHR$(17)
[CRSR LINKS]		=	<SHIFT>+<CRSR>	CHR$(157)
[CRSR RECHTS]		=	<CRSR>	CHR$(29)
[SCHWARZ]		=	<CTRL>+<1>	CHR$(144)
[WEISS]		=	<CTRL>+<2>	CHR$(5)
[ROT]		=	<CTRL>+<3>	CHR$(28)
[CYAN]		=	<CTRL>+<4>	CHR$(159)
[PURPUR]		=	<CTRL>+<5>	CHR$(156)
[GRUEN]		=	<CTRL>+<6>	CHR$(30)
[BLAU]		=	<CTRL>+<7>	CHR$(31)
[GELB]		=	<CTRL>+<8>	CHR$(158)
[REVERS EIN]		=	<CTRL>+<9>	CHR$(18)
[REVERS AUS]		=	<CTRL>+<0>	CHR$(146)
[ORANGE]		=	<C=>+<1>	CHR$(129)
[BRAUN]		=	<C=>+<2>	CHR$(149)
[HELLROT]		=	<C=>+<3>	CHR$(150)
[GRAU 1]		=	<C=>+<4>	CHR$(151)
[GRAU 2]		=	<C=>+<5>	CHR$(152)
[HELLGRUEN]		=	<C=>+<6>	CHR$(153)
[HELLBLAU]		=	<C=>+<7>	CHR$(154)
[GRAU 3]		=	<C=>+<8>	CHR$(155)

<CTRL> BEDEUTET 'CONTROL-TASTE'.
<C=> BEDEUTET 'COMMODORE-TASTE'.

VORWORT

Dieses Handbuch und die Serie, zu dem es gehört, wurde geschrieben, um eine offensichtliche Lücke in den Handbüchern für Besitzer von Homecomputern zu schließen. Fehlten doch Bücher, die den Traum eines jeden Computerbesitzers wahr machen, daß die neue Maschine nämlich nicht nur ein Spielzeug oder eine Einführung in das Silikon-Zeitalter
darstellt, sondern ein Hilfsmittel, das die verschiedensten Aufgaben übernimmt und eine Vielzahl von Möglichkeiten eröffnet. Die meisten auf dem Markt befindlichen Bücher befassen sich entweder mit Allgemeinplätzen oder gehen von dem Wunsch - vielleicht sogar von der Möglichkeit - zum Experimentieren aus.
Ich wollte ein Buch schreiben, das auf einer soliden Sammlung von Programmen beruht, die eigentlich jeder haben möchte - Programmen, die Bereiche wie Datenspeicherung, Finanzverwaltung, Grafik, Musik, Haushaltsverwaltung und Ausbildung abdecken. Die Besprechung der Programmiertechniken ergibt sich aus den Programmen selbst und nicht aus einer Liste von Dingen, die erlernt werden müssen. Ich hoffe, daß Sie dieses aus diesem Wunsch entstandene Buch als hilfreich empfinden werden. Es soll nicht nur als Möglichkeit gesehen werden, neue Programmiertechniken zu erlernen, sondern auch als eine Sammlung von Programmen selbst, die alle unabhängig auf Fehler geprüft wurden und eine Vielzahl von Anwendungen bieten, die bislang nur Benutzern zur Verfügung standen, die bereit waren, eine teure kommerzielle Software zu erwerben, oder die in der Lage waren, hochentwickelte Programme für den eigenen Bedarf zu schreiben.
Neben den Programmen in diesem Handbuch verfügen Sie über die einzelnen Elemente der Programme - denn die Programme in diesem Handbuch sind in 'modularer Form' geschrieben. Dies bedeutet, daß sie aus eindeutig identifizierbaren funktionalen Einheiten bestehen, die sobald sie einmal verstanden wurden, aus dem Programm herausgenommen und für den eigenen Zweck eingesetzt werden können. Bei jedem Modul werden die neue Aspekte genau kommentiert. Außerdem wird beschrieben, wie die Programme getestet werden, nachdem die einzelnen Moduln eingegeben wurden.
Sie werden in diesem Handbuch zwar auch auf Abschnitte stoßen, in denen allgemeine Themen besprochen werden. Sie werden jedoch feststellen, daß es sich hier nicht um ein Buch handelt, das gelesen, sondern das benutzt werden soll. Die Relevanz der Kommentare und Empfehlungen wird erst deutlich, wenn die auf den ersten Blick sehr langen und komplexen Programme einmal eingegeben werden. Hier verhindert die modulare Lösung, daß Programme zu verwirrenden Fehlerquellen werden, so daß unbedingt empfohlen wird, die Moduln wie vorgeschlagen zu testen, insbesondere im Anfangsstadium.
Schließlich muß der Erfolg oder Mißerfolg dieses Buches jedoch danach beurteilt werden, ob mit ihm die Arbeit mit dem Commodore 64 einfacher wird oder nicht.

Dieses Buch ist in erster Linie ein Buch für den C64. Wenn auch die allgemeine Struktur des Handbuchs auf den beiden Vorgängern in dieser Serie beruht, so wurden die Programme doch angepaßt und neue Programme hinzugefügt, um die besonderen Möglichkeiten des C64 voll auszuschöpfen. Wenn es auch eine harte Arbeit ist, ein Buch wie dieses zu schreiben, war es doch schön, zu sehen, was der C64 aus Programmen gemacht hat, die bei weniger leistungsfähigen Systemen wesentlich weniger aufregend sind. Die Benutzung dieser Programme ist nicht ganz so arbeitsaufwendig - das Ergebnis ist jedoch genau so aufregend.

Schließlich darf keine Einführung in ein Buch wie dieses ohne Dank an Commodore UK für die zur Verfügung gestellten Einrichtungen abgeschlossen werden. Nicht zuletzt bin ich Steve Beats in der Hauptverwaltung von Commodore UK für seine Geduld bei der Beantwortung der dümmsten Fragen zu Dank verpflichtet, die sich bei der Benutzung des C64 für mich eröffneten.

David Lawrence

Nach dem großen Erfolg, den David Lawrence mit seinem Buch in England gehabt hat, bringen wir auch für den deutschen Leser dieses Standardwerk heraus, das für das Programmieren nicht nur am Commodore 64 sondern ebenso auch am 'größeren Bruder', dem neuen Commodore 128, seine Aktualität behalten wird.

Commodore Verlagsabteilung
Sachbuchredaktion

KAPITEL 1

KLEIN ABER OHO

Die Programme in diesem Handbuch sollen mit einer Vielzahl wichtiger Anwendungen eingesetzt werden. Da viele der Anwendungen komplex sind, gilt dies auch für viele der Programme. Dies bedeutet jedoch nicht, daß nützliche Programme nicht auf kleinem Raum komprimiert werden können. Als Einführung in die gewählte Lösung werden in diesem Kapitel drei relativ kurze Programme vorgestellt, die alles andere als ein Spielzeug sind.

1.1 Clock

Dieses Programm ist eine recht amüsante Einführung in einige der Funktionen des Commodore 64 - es ist problemlos einzugeben, macht Spaß, wenn es auf dem Bildschirm vor der ganzen Familie vorgeführt wird, und nutzt die Möglichkeiten des C64 für die flexible Zeichenfolgen- und Bildschirmbearbeitung voll aus.
Das Programm erfüllt genau die Funktion seines Namens, einer Uhr. Allerdings werden bei der Ausführung des Programms weder ein Ziffernblatt noch ein Zeiger sichtbar. Die 64 Uhr benutzt zwei über den Bildschirm laufende Linien, die eine von links nach rechts für die Minuten und die andere von oben nach unten für die Stunden. Auf diese Weise wird der Bildschirm in verschiedene Farbbereiche unterteilt. All dies ist nur möglich, da der C64 über eine flexible Zeitfunktion verfügt, die auf direktem Wege aus dem Programm heraus festgelegt und gelesen werden kann.

Clock: Variablentabelle

CS	Anfangsadresse des Farbspeichers
DT$	Formatierte Anpassung von TI$
H	Stundenwert, in Bildschirmeinheiten unterteilt
M	Minutenwert, in Bildschirmeinheiten unterteilt
M1$,M2$	Zweifarbige Zeichenfolgen, mit denen die Stunden- und Minutenwerte angezeigt werden
SS	Anfangsadresse des Bildschirms
TI$	Eine Systemvariable, die die Zeit nach dem internen Takt enthält.

MODUL 1.1.1

```
11000 REM#*******************************
*****
11010 REM UHR STELLEN UND BILDSCHRIM AUF
BAUEN
11020 REM*******************************
*****
11030 POKE 53280,0:POKE 53281,1
11040 PRINT "▟[GRUEN]▩BITTE GEBEN SIE DI
E STUNDEN EIN (00-23):";:INPUT H$
11050 PRINT "▩[ROT]BITTE GEBEN SIE DIE M
INUTEN EIN (00-59):":INPUT M$
11060 TI$=H$+M$+"00"
11070 PRINT "▟[REVERS EIN][SCHWARZ]     5
 10 15 20 25 30 35 40 45 50 55 60   "
11080 SS=1024:CS=55296:FOR I=0 TO 24
11090 POKE CS+40*I,0:POKE SS+40*I,160
11100 POKE CS+40*I+1,0:POKE SS+40*I+1,16
0
11110 POKE CS+40*I+38,0:POKE SS+40*I+38,
160
11120 POKE CS+40*I+39,0:POKE SS+40*I+39,
160
11130 NEXT I
11140 PRINT "▩":FOR I=1 TO 11:PRINT "[RE
VERS EIN]";MID$(STR$(I),2):PRINT:NEXT
11150 PRINT "[REVERS EIN]";MID$(STR$(I),
2);
```

Mit diesem Modul kann der Benutzer die Zeit in Stunden und Minuten (12-Stunden-Takt) eingeben. Mit ihm wird der Zeitgeber festgelegt und danach die Uhrzeit angezeigt.

Kommentar

Zeile 11030: Zwei besonders nützliche Speicheradressen: 53280 - definiert die Farbe der Bildschirmumrahmung neu, 53281 definiert die Farbe des Bildschirmhintergrunds neu. Beide können während der Ausführung eines Programms sofort zurückgesetzt werden. In diesem Fall wird die Umrahmung wieder schwarz und der Bildschirmhintergrund wieder weiß angezeigt.

Zeilen 11040-11060: Die Stunden und Minuten werden in Form von zweistelligen Zahlen eingegeben. Danach werden sie addiert und wird 00 für die Sekunden hinzugefügt. Dem System wird mitgeteilt, daß dies TI$ ist. Es setzt den internen Takt dann sofort so zurück, daß ab dieser Zeit gezählt wird.

Zeile 11070: Der Bildschirm wird gelöscht. Die Bildschirmfarbe wird auf schwarz festgelegt. Außerdem wird die inverse Anzeige festgesetzt. Danach werden die Zahlen am Anfang des Bildschirms ausgedruckt.

Zeilen 11080-11130: Die schwarzen Umrahmungen des Uhrzeitbereichs werden nun an den Rand des Bildschirms gesetzt. Wird direkt am Bildschirmrand gedruckt, so ist es häufig bequemer, Zeichen mit POKE auf den Bildschirm zu setzen, da dadurch verhindert wird, daß die Druckposition von einer Zeile zur nächsten springt. Um den Bildschirm erfolgreich mit POKE zu füllen, muß mit zwei Adressen gearbeitet werden, einer direkt innerhalb des Bildschirmspeichers selbst (Adressen 1023-2023) und einer anderen innerhalb des Farbspeichers (55296-56295). Mit dieser Schleife werden die ersten beiden Zeichen und die letzten beiden Zeichen der 25 Zeilen auf dem Bildschirm mit einem Zeichencode von 160 (einem inversen Leerzeichen) und die entsprechende Adresse im Farbspeicher mit Null gefüllt, wodurch diese Zeichenposition schwarz angezeigt wird.

Zeile 11140: Der Cursor wird wieder in die Ausgangsposition bewegt und die Stunden werden entlang der linken Seite des Bildschirms angezeigt. Der letzte Wert wird separat mit einem abschließenden Semikolon ausgedruckt, so daß der Bildschirm nicht nach oben läuft, da auf der untersten Bildschirmzeile gearbeitet wird.

Testen von Modul 1.1.1

Hierzu wird eine temporäre Zeile 11160 GOTO 11160 eingefügt und das Modul ausgeführt. Sie werden aufgefordert, die Stunden und Minuten einzugeben. Danach wird die Umrahmung der Uhr des Zifferblattes auf dem Bildschirm angezeigt. Durch die temporäre Zeile wird gewährleistet, daß der Bildschirm nicht nach oben läuft und READY ausdruckt, sobald das Modul beendet ist.

MODUL 1.1.2

```
12000 REM#*******************************
*
12010 REM BERECHNUNG UND AUSGABE DER ZEI
T
12020 REM********************************
```

```
*
12030 M=INT((VAL(MID$(TI$,3,2))+0.8)*3/5
)
12040 H=2*VAL(MID$(TI$,1,2))
12050 IF M>=30 THEN LET H=H+1
12060 IF H>=24 THEN LET H=H-24
12070 IF M=0 THEN LET M=1
12080 M1$="■■[PURPUR][REVERS EIN]
                                   "

12090 LET M1$=LEFT$(M1$,M+4)+"[ROT]"+RIG
HT$(M1$,36-M)
12100 M2$="■■[BLAU][REVERS EIN]
                                 "

12110 LET M2$=LEFT$(M2$,M+4)+"[WEISS]"+R
IGHT$(M2$,36-M)
12120 PRINT "■■";
12130 IF H>0 THEN FOR I=1 TO H:PRINT M1$
:NEXT
12140 IF H<23 THEN FOR I=H+1 TO 23:PRINT
 M2$:NEXT
12150 PRINT M2$;
12160 PRINT "[SCHWARZ]■■■■■■■■■■■■■■■■■■
                                   ";

12170 DT$=LEFT$(TI$,2)+"   "+MID$(TI$,3,2
)+"   "+RIGHT$(TI$,2)
12180 FOR I=1 TO LEN(DT$):PRINT MID$(DT$
,I,1);"■■";:NEXT
12190 GOTO 12030
```

Mit diesem Modul werden die Werte abgeleitet, anhand derer der interne Takt angezeigt werden kann. Außerdem wird die Zeit in zwei Formen angezeigt.

Kommentar

Zeile 12030: Auf dem Bildschirm stehen 36 Leerzeichen zur Verfügung, nachdem die Umrahmungen gezeichnet wurden. Auf diese Weise muß die Anzahl von Minuten durch 60/36 (5/3) dividiert werden, um die richtigen Einheiten zu erhalten, mit denen über den Bildschirm gegangen werden kann.

Zeile 12040: Auf dem Bildschirm stehen 24 Zeilen zur Verfügung, so daß die Stunden nur mit 2 multipliziert zu werden brauchen.

Zeile 12070: Das Programm zeigt stets Minuten an, so daß bei der vollen Stunde der Minutenwert erhöht wird, um eine Einheit anzugeben.

Zeile 12080: M1$ ist gleichbedeutend mit zwei Cursorbewegungen nach rechts plus dem dunkelroten Steuerzeichen und dem Steuerzeichen für die inverse Anzeige, gefolgt von 36 Leerzeichen. Beim Ausdrucken ergibt dies eine purpurfarbene Linie.

Zeile 12090: M1$ wird nun geändert, so daß es gleichbedeutend mit den vier ersten Steuerzeichen plus M Leerzeichen, einem roten Steuerzeichen und dann den restlichen Leerzeichen ist. Dadurch wird eine neue Zeichenfolge erstellt, die die Farbe an einem durch den Wert von M definierten Punkt ändert.

Zeilen 12100-12110: Dasselbe Verfahren wird für M2$ ausgeführt, das blau beginnt und weiß endet.

Zeilen 12130-12150: M1$ wird für so viele Zeilen ausgedruckt, wie Stunden vorhanden sind; M2$ wird auf den restlichen Zeilen ausgedruckt. Die beiden Zeichenfolgen definieren so einen Rahmen zwischen verschiedenen Farbbereichen, die dem Wert von H unterliegen.

Zeile 12160: Der Cursor wird in die Ausgangsposition bewegt. Die Druckposition wird um etwa ein Drittel in der vorletzten Spalte des Bildschirms nach unten bewegt, wobei die Druckfarbe auf schwarz festgesetzt wird.

Zeile 12170: DT$ ist nun als TI$ definiert, wobei zwischen die Stunden-, Minuten- und Sekundenwerte zwei Leerzeichen eingefügt werden. Während des Programms hat das System TI$ ständig aktualisiert, so daß es stets die laufende Zeit enthält.

Zeile 12180: DT$ wird entlang der rechten Seite des Bildschirms ausgedruckt. Hier wird jeweils ein Zeichen ausgedruckt und der Cursor dann nach unten und zurück bewegt.

Testen von Modul 1.1.2

Nun sollte die Uhr laufen, wobei vier verschiedene Farbrechtecke die Linien für Stunden und Minuten markieren. Die Zeit wird digital auf der rechten Seite des Bildschirms angezeigt.
Trotz der Beschreibung sind höchstwahrscheinlich in der vorgenommenen Eingabe einige Fehler enthalten. Ist dies bei diesem Programm noch nicht der Fall, so wird es bei späteren Programmen so sein. Anhand der Anfragen glaube ich,

daß die meisten Besitzer von Homecomputern am Anfang Probleme haben, derartige Fehler zu beheben. Vielleicht sind hier einige grundlegende Richtlinien von Hilfe:

1. Nutzen Sie die Ihnen zur Verfügung stehende Hilfe maximal. Wird eine Fehlermeldung angezeigt, so berücksichtigen Sie diese und halten die Zeile fest, in der der Fehler aufgetreten ist, sowie die Art des Fehlers.
2. Das Programm sollte nicht noch einmal ausgeführt werden, um zu sehen, ob der Fehler auch beim zweiten Mal auftritt. Läuft dann nämlich alle glatt, haben Sie die Chance verpaßt, den Fehler sofort zu beheben.
3. Benutzen Sie den direkten Modus (Befehle werden direkt über die Tastatur und nicht über Programmzeilen eingegeben), um die Werte sämtlicher Variablen in den Zeilen auszudrucken, die einen Fehler aufzuweisen scheinen. Ein seltsamer Wert zeigt häufig den Weg zu dem Fehler. Eine große Anzahl von nahezu nicht feststellbaren Fehlern sind einfach auf die falsche Eingabe eines Variablennamens zurückzuführen, beispielsweise auf die Eingabe von I für 1.
4. Vollziehen Sie das Programm im Kopf oder auf Papier nach, wobei einfache Werte benutzt werden, so daß Sie genau sehen, was das Programm an dieser Stelle tun muß.
5. Nehmen Sie Änderungen nicht zu schnell vor, bevor Sie sich nicht sicher sind, daß es sich hier um die einzige Änderung handelt, die vorgenommen werden soll. Nachdem Sie eine Änderung in eine Zeile eingeben, verschwinden sämtliche Daten und somit die Möglichkeit, weitere Prüfungen vorzunehmen, ohne das Programm erneut ausführen zu müssen.
6. Saven Sie das Programm regelmäßig, wenn Sie Fehler feststellen und/oder neue Zeilen hinzufügen. Viele Fehler in den endgültigen Programmen ergeben sich aus Änderungen, die in ein Programm eingegeben, jedoch niemals auf Band festgehalten wurden. Alle meine Programme beginnen mit den drei folgenden Zeilen:

```
1 GOTO 3
2 SAVE "XXXX":STOP
3 REM
```

Mit diesen drei Zeilen können Programme mit dem Befehl 'GOTO 2' gesaved werden (vorausgesetzt 'XXXX' wird durch den Programmnamen ersetzt). Daraus ergibt sich außerdem der Vorteil, daß ich meine Programme stets mit GOTO 1 starten kann, anstatt mich an die erste Zeilennummer des Hauptprogramms erinnern zu müssen.
Beim Entwurf und bei der Eingabe von Programmen macht jederman Fehler. Der Unterschied besteht einfach darin, ob man es lernt, mit diesen Fehlern fertig zu werden oder nicht.

Zusammenfassung

Ob Sie die Uhr nun mögen, können nur Sie beurteilen. Ich persönlich finde das Programm recht attraktiv. Unabhängig von der Uhr sind die Techniken der Aufteilung von Zeichenfolgen und der Änderung des Bildschirm und des Farbspeichers innerhalb des Programms mit POKE in einer Vielzahl von Programmen sehr nützlich, so daß es sich durchaus lohnt, das Programm einzugeben und zu verstehen, wie es funktioniert.

1.2 GRAPH

Möchten Sie dieses Programm verstehen, so können Sie nichts besseres tun, als den Karton zu betrachten, in dem der Commodore 64 geliefert wurde. Dort sehen Sie ein farbiges, dreidimensionales Balkendiagramm. Mit dem vorliegenden Programm soll versucht werden, das Programm zu reproduzieren, mit dem dieses Diagramm erstellt wurde. Ich sage versucht, da ich nach erfolgreicher Reproduktion des Diagramms auf dem Karton festgestellt habe, daß die Daten, mit denen gearbeitet wurde, sorgfältig ausgewählt wurden, um die Grenzen der eng gepackten Balken zu verdecken. Andere Daten führten zu Grafikzeichen, bei denen die Balken Löcher in die benachbarten Balken schlagen, so daß das Ganze weit weniger attraktiv aussieht, als auf dem Karton dargestellt.
Dieses Programm ist also ein Kompromiß, der zwar eine weniger eng gepackte Anzeige erzeugt, jedoch mit beliebigen Daten benutzt werden und dennoch gut aussehen kann - so gut, daß Sie nach Beendigung der Eingabe Ihre Familie rufen werden, um sie mit Ihrem Können zu beeindrucken.
So farbig und praktisch werden die erzeugten Anzeigen fraglos viele Anwendungen finden. Darüber hinaus bietet das Programm eine einfache Einführung in die Sicherung von Daten auf Magnetband und das spätere erneute Laden dieser Daten.

Graph: Variablentabelle

CO$	Aus drei Zeichen bestehende Zeichenfolge, mit der die Farbe der verschiedenen Balken in dem Diagramm festgelegt wird.
F$	Formatierfolge für Cursorzeichen nach rechts.
F1$,F3$	Formatierfolgen für Cursorzeichen nach unten.
F2$	Aus F$ abgeleitete temporäre Zeichenfolge.
HH(2,6)	Matrix mit Daten für die Grafik.
NB	Anzahl von voreinander stehenden Reihen (1-3).
ND	Anzahl von Spalten entlang der Horizontalachse (1-6).
NH$	Name der Horizontalachse.
NV$(2)	Namen für jede separate Reihe.

TT$ — Temporäre Zeichenfolge, mit der das Ausdrucken der Namen der vertikalen Achse formatiert wird.

UV — Zahl, die von jeder Einheit auf der vertikalen Achse dargestellt werden soll.

MODUL 1.2.1

```
11000 REM#****************************
11010 REM DATENEINGABE
11020 REM*****************************
11030 POKE 53281,15:INPUT "[SCHWARZ]WO
LLEN SIE VON KASSETTE LADEN (J/N):";Q$
11040 IF Q$="J" THEN 12420
11050 PRINT "[REVERS EIN][
ROT]GRAPH"
11060 PRINT "[SCHWARZ]ES GIBT 19 VERTIK
ALE EINHEITEN."
11070 PRINT "[BLAU]BITTE GEBEN SIE DAS
MASS EINER EINHEIT  EIN."
11080 INPUT "MASS:";UV
11090 INPUT "[ROT]NAME DER HORIZONTALEN
 ACHSE:";NH$
11100 PRINT "[SCHWARZ]SIE KOENNEN EINS
 BIS SECHS SPALTEN       WAEHLEN."
11110 INPUT "WIEVIELE MOECHTEN SIE :";N
D
11120 PRINT "[GRUEN]BIS ZU DREI REIHEN
 SIND MOEGLICH."
11130 INPUT "WIEVIELE MOECHTEN SIE :";N
B
11140 FOR I=0 TO NB-1
11150 PRINT "[GRUEN]NAME DER VERTIKALEN
 ACHSE";I+1;:INPUT NV$(I):NEXT I
11160 DIM HH(2,6)
11170 PRINT "";:FOR I=0 TO NB-1
11180 FOR J=1 TO ND
11190 PRINT "WERT FUER REIHE";I+1;" SPA
LTE";J;":";:INPUT T
11200 IF INT(T/UV)>19 THEN PRINT "WERT
ZU GROSS.":GOTO 11190
11210 HH(I,J)=T:NEXT J,I
```

Mit diesem sehr direkten Modul soll die Eingabe von Daten ermöglicht werden, die für das Erstellen und die Beschriftung der Grafik benutzt werden. Anstatt den Benutzer aufzufordern, die Höchst- und Mindestzahlen für den Wertebereich einzugeben und danach die Einheiten zu berechnen (was zu äußerst seltsamen Einheiten führen kann), muß der Benutzer bei diesem Programm angeben, wieviele Einheiten der eingegebenen Daten von jeder vertikalen Einheit in der Grafik dargestellt werden sollen. Der Horizontalachse und den Reihen mit den dreidimensionalen Pfeilern werden Namen zugewiesen, wobei von hinten begonnen wird. Schließlich werden Daten je nach der vom Benutzer ausgewählten Struktur angefordert.

Testen von Modul 1.2.1

Hier braucht nur das Modul ausgeführt zu werden, um die Syntax auf Richtigkeit zu überprüfen. Bis jetzt kann noch nichts gezeichnet werden.

MODUL 1.2.2

```
12000 REM#*******************************
12020 REM GRAPH ZEICHNEN
12030 REM********************************
12040 POKE 53281,0:PRINT "[BRAUN]
";
12050 F$=""
12060 FOR I=1 TO 4:PRINT F$;"[REVERS EI
N]                                 ":F$=F$+
"":NEXT
12070 PRINT "[WEISS]______________
_______________":REM ZEILE MIT 30 ZEICH
EN
12080 FOR I=1 TO 19:PRINT "_";
12090 PRINT "
 _|":NEXT
12100 PRINT "";:FOR I=1 TO 4:PRINT "
     ";
12110 PRINT "__________________________
_":NEXT:REM ZEILE MIT 28 ZEICHEN
12120 F1$="":F$="
                              ":CO$="[GELB
][GRUEN][PURPUR]"
12130 PRINT "";
```

```
NH$;
12140 FOR H=0 TO NB-1:PRINT "████████████";
MID$(F$,1,2*(H+1));MID$(CO$,H+1,1);
12150 TT$=NV$(H)+" *"+STR$(UV):FOR I=1 T
O LEN(TT$)
12160 PRINT MID$(TT$,I,1);"██";:NEXTI,H
12170 PRINT "████████████15████████10████████5
0"
12180 F3$=F1$:FOR H=0 TO NB-1:PRINT MID$
(CO$,H+1,1)
12190 F2$=LEFT$(F$,8+4*(ND-1)+H):PRINT F
1$;F2$;
12200 FOR I=ND TO 1 STEP-1:IF INT(HH(H,I
)/UV)=0 THEN 12270
12210 FOR J=1 TO INT(HH(H,I)/UV)+H
12220 IF INT(HH(H,I)/UV)=0 THEN 12270
12230 IF J=1 THEN PRINT "[BRAUN][REVERS
EIN] ◥";MID$(CO$,H+1,1);"|  [REVERS AUS]
";
12240 IF J>1 THEN PRINT " ▌[REVERS EIN]▌
 [REVERS AUS]";
12245 REM ZEICHEN SIND " ▌.  .[REVERS EIN]
.▌.  .[REVERS AUS]....";
12250 NEXT J
12260 PRINT " ◤[REVERS EIN] ◥"
12270 PRINT:F2$=LEFT$(F2$,LEN(F2$)-4):PR
INT F1$;F2$;:NEXT I
12280 F1$=F1$+"█"
12290 F2$=LEFT$(F$,9+4*(ND-1)):PRINT F3$
;F2$;
12300 NEXT H
12310 FOR I=1 TO ND:PRINT F3$;
12320 F2$=LEFT$(F$,9+4*(I-1)):PRINT F2$;
:FOR J=1 TO NB
12330 IF J>1 OR HH(2,I-1)=0 THEN PRINT "
[BRAUN][REVERS EIN]◥";
12340 PRINT "███";:NEXT J,I
12350 GET A$:IF A$="" THEN 12350
```

Dies ist ein besonders leistungsfähiges Modul, das nicht auf einer klaren Gruppe von Methoden aufgebaut ist, sondern einfach auf den Bedingungen, die meiner

Meinung nach in der Praxis erfüllt werden müssen, um eine attraktive Darstellung der Grafik zu erzielen.

Kommentar

Zeilen 12040-12060: Der Bildschirmhintergrund ist auf die schwarze Farbe festgelegt. Die braune Grundlage, auf der die Grafik steht, wird eingezeichnet.

Zeilen 12070-12110: Das Raster, das den Grafikbereich umgibt, wird gezeichnet. Einheiten werden an den Seiten markiert und Linien werden quer gezeichnet, um die fünf Einheitenebenen zu markieren.

Zeilen 12140-12160: Mit diesen drei Zeilen wird eine Druckposition in der oberen rechten Ecke des Bildschirms festgelegt. Außerdem werden die Namen der drei Reihen entlang dem Bildschirm in den Farben ausgedruckt, die den Reihen selbst entsprechen. Die Bildschirmposition wird anhand eines Datenpostens aus F$ bestimmt. Die Farbe wird bestimmt, indem jeweils ein anderes der drei Steuerzeichen für die Farbe bei jeder Ausführung der Schleife ausgedruckt wird.

Zeilen 12180-12300: In diesem Abschnitt werden drei Schleifen aufgerufen. Mit der Schleife H wird festgelegt, wie viele Reihen voreinander gezeichnet werden. Mit ihr wird auch ein Steuerzeichen für die Farbe aus CO$ ausgegeben und bestimmt, wie viele Cursorzeichen nach unten ausgedruckt werden, wobei die Reihen dementsprechend nach unten bewegt werden. Mit der Schleife I wird festgelegt, wie viele Spalten über die Bildschirmbreite ausgedruckt werden. Mit der Schleife J wird festgelegt, wie hoch ein bestimmter dreidimensionaler Block ist.

Zeile 12190: Die horizontale Druckposition am Anfang jede Reihe (die Reihen werden von rechts nach links gezeichnet) wird anhand der Reihe berechnet - jede Reihe wird über eine andere Reihe bewegt, so daß die drei Reihen ein dreidimensionales Aussehen erhalten.

Zeile 12200: Eine Spalte wird nicht gedruckt, wenn das entsprechende Matrixelement in der Matrix HH gleich Null ist.

Zeilen 12230-12240: Am Ende jeder Spalte wird das schräge Ende ausgedruckt, durch das die Spalte so aussieht, als stünde sie auf der Fläche.

Zeile 12260: Ist die Spitze der Spalte erreicht, so wird die schräge Spitze hinzugefügt.

Zeile 12270: Bei der nächsten Spalte werden vier Zeichen von der Zeichenfolge

für den Cursor nach rechts subtrahiert, so daß die neue Druckposition definiert wird.

Zeile 12280: Die vertikale Druckposition wird nach unten bewegt, bevor die nächste Reihe ausgedruckt wird.

Zeilen 12310-12340: Beim Ausdrucken der Reihen wurden die unteren Enden der Spalten zerstört. Sie werden jetzt ausgefüllt.

Zeile 12350: Die Grafik bleibt auf dem Bildschirm angezeigt, bis eine Taste betätigt wird.

Testen von Modul 1.2.2

Auch hier wieder ein einfaches Ausführen des Programms, bei dem geprüft wird, ob die Bildschirmanzeige richtig aussieht. Sollte es zu Problemen kommen, so wird die Anzahl von Reihen, und vielleicht sogar die Anzahl von Spalten, auf Eins beschränkt, um die Fehleranalyse zu vereinfachen. Die Funktion jeder der Schleifen wird separat betrachtet, um festzustellen, welche der Schleifen den Fehler erzeugt. Dieses Modul ist relativ schwierig auszutesten. Sind Sie also der Meinung, daß das Problem durch eine Änderung in einer der Zeilen gelöst werden könnte, auch wenn Sie nicht feststellen können, an welcher Stelle Sie von der oben angeführten Liste abgewichen sind, so nehmen Sie die Änderung ruhig vor und prüfen das Ergebnis - nichts an dieser oder einer anderen Stelle in diesem Handbuch ist heilig.

MODUL 1.2.3

```
12360 INPUT "[WEISS]WOLLEN SIE DIE DAT
EN SPEICHERN (J/N):";Q$:IF Q$="N" THEN E
ND
12370 INPUT "KASSETTE EINLEGEN, DANN [R
EVERS EIN]RETURN[REVERS AUS] --";Q$:R$=C
HR$(13)
12380 OPEN 1,1,1,"GRAPH"
12390 PRINT#1,NB;R$;ND;R$;NH$;R$;UV
12400 FOR I=0 TO NB-1:PRINT#1,NV$(I):FOR
 J=0 TO ND:PRINT#1,HH(I,J):NEXT J,I
12410 CLOSE1:END
12420 INPUT "[GRUEN]KASSETTE EINLEGEN,
 DANN [REVERS EIN]RETURN[REVERS AUS] --"
;Q$:DIM NV$(2),HH(2,6)
```

```
12430 OPEN 1,8,0,"GRAPH"
12440 INPUT#1,NB,ND,NH$,UV
12450 FOR I=0 TO NB-1:INPUT#1,NV$(I):FOR
 J=0 TO ND:INPUT#1,HH(I,J):NEXT J,I
12460 CLOSE1:GOTO 12000
12470 GOTO 12470
```

Nachdem nun die Grafik definiert wurde, können die Daten auf Band gespeichert werden, anstatt sie zu verlieren und später neu eingeben zu müssen. Mit diesem Modul können die Daten gesichert und später wieder aufgerufen werden. Dieses Modul soll die Bandspeicherung insofern so einfach wie möglich gestalten, als es Ihnen die Zeit gibt, das Band richtig einzulegen, bevor mit dem Laden oder Sichern begonnen wird.

Kommentar

Zeile 12380: Mit dieser Zeile wird eine Datei geöffnet, in die die Daten gesetzt werden sollen. In diesem Fall ist der Speicherplatz ein Kassettenrecorder. Die drei Zahlen stellen folgende Angaben dar:

a) Die Nummer der Datei - bei jeder Anweisung, Daten in einer Datei zu speichern, muß die Dateinummer angegeben werden.
b) Die Einheitennummer (das Anlagenteil, in dem die Daten empfangen werden sollen), wobei 1 den Kassettenrecorder darstellt.
c) Den Dateityp - 1 bedeutet, daß es sich hier um eine Datei handelt, in die Daten gespeichert werden, und nicht um eine Datei, aus der Daten genommen werden.

NB, ND, NH$ und UV werden in die Datei (auf Band) gespeichert. Hier sollte die Benutzung der Variablen R$ beachtet werden. Bei der Speicherung von Daten auf Band ist der C64 etwas anspruchsvoll, was die Trennung der einzelnen Elemente voneinander betrifft. Werden sie einfach durch Kommas voneinander getrennt, so kann dies zu Fehlern führen, wenn die Daten erneut geladen werden. R$ wurde in Zeile 12370 als CHR$(13) definiert, den Code für RETURN. Wird dieser Code zwischen die einzelnen Elemente gesetzt, so ist gewährleistet, daß sie richtig voneinander
getrennt sind.

Zeile 12400: Die einzelnen Feldelemente werden nacheinander in die Datei gedruckt.

Zeile 12410: Nachdem die Speicherung von Daten in einer Datei beendet ist, muß die Datei mit CLOSE geschlossen werden. Geschieht dies nicht, so kommt es

beim nächsten Öffnen einer Datei mit derselben Nummer zu einem Fehler.

Zeile 12420: Dies ist der Teil des Moduls, mit dem die Daten wieder in den C64 geladen werden. Der einzige Unterschied zwischen den Spezifikationen für die beiden Dateitypen besteht darin, daß diese Datei den Dateityp 0 aufweist. Dies bedeutet, daß es sich hier um eine Datei handelt, aus der Daten genommen werden.

Zeilen 12440-12450: Die Daten, die in die Datei eingegeben wurden, werden nun wieder aufgerufen. Der sicherste Weg für das Erstellen einer Laderoutine besteht in der Änderung der Zeilennummern der SAVE-Routine und in der Änderung der PRINT-Befehle in INPUT-Befehle. Auf diese Art und Weise wissen Sie, daß die Routine die Daten genau in der Reihenfolge wiederaufnimmt, in der sie gespeichert wurden. Werden die Daten in der falschen Reihenfolge herausgenommen, so führt dies nicht nur zu einem unsinnigen Programm, sondern ein Fehler kann auch zu einem Programmstopp führen.

Testen von Modul 1.2.3

Bei diesem Modul wird einfach getestet, ob Daten in das Programm eingegeben, auf Band gesichert und dann wieder geladen werden können.

Zusammenfassung

Dieses Programm ist ein Tribut an die Grafikmöglichkeiten und die Möglichkeiten der Bildschirmbearbeitung des Commodore 64. Nach der Eingabe werden Sie feststellen, daß es gar nicht so schwierig ist, Schleifen und einfache Berechnungen dazu zu benutzen, Formen und offensichtlich feste Gegenstände an bestimmten Stellen auf dem Bildschirm zu zeichnen. Außerdem werden Sie feststellen, daß mit dieser Art der Anzeige Informationen mit Abstand am wirksamsten übermittelt werden können.

1.3 TEXTED

Das letzte Programm in diesem Kapitel ist ein Versuch, einige der einfacheren Funktionen eines Textverarbeitungssystems in einem relativ kurzen und unkomplizierten Programm zu vermitteln. Das Programm hält selbstverständlich den Vergleich mit einem professionell geschriebenen, auf dem Maschinencode basierenden Textverarbeitungssystem nicht aus und ich hätte es auch nicht dazu benutzt, ein Buch wie dieses zu schreiben. Und sei es nur, weil Commodore UK mir eine Kopie des erstklassigen Easyscript Programms für diesen Zweck zur

Verfügung gestellt hat. Dennoch, das Programm funktioniert, und ich würde und habe es für viele Zwecke einer Schreibmaschine vorgezogen, da es bei der Eingabe und Bearbeitung von Texten vor der endgültigen Ausgabe auf Papier überaus flexible Möglichkeiten bietet. Verfügen Sie natürlich über keinen Drucker, so müssen Sie sich schnell einen besorgen, um das Programm maximal ausnutzen zu können.

Texted: Variablentabelle

A$	Eingegebene Textzeile
CH	Zeichencode unter dem blinkenden Cursor
FNA(P)	Berechnet die Speicherposition des blinkenden Cursors in dem gerade eingegebenen Text
FNB(P2)	Berechnet die Speicherposition des Bearbeitungs-Cursors im Haupttext
LL	Anzahl von Textzeilen im Haupttext
P	Position des blinkenden Cursors in der gerade eingegebenen Zeile
P2	Position des Bearbeitungscursors auf dem Bildschirm
PL	Zeilennummer des Bearbeitungs-Cursors im Haupttext
SP	Anzahl von Leerstellen, die beim Formatieren am Zeilenende zur Verfügung stehen
SS	Zeilennummer der ersten Zeile des gerade angezeigten Teils des Haupttextes
T$	Letztes eingegebenes Zeichen bei der Eingabe einer neuen Textzeile
T1$	Als Befehl in dem Modul zur Verschiebung der Bearbeitungszeile eingegebenes Zeichen
TEXT$(500)	Hauptmatrix für die Speicherung des Textes
TT$	Vorübergehender Speicherplatz für Zeilen, die in den Haupttext eingegeben werden
X	Anzahl von Textzeilen, die aus dem Textstapel ausgegeben werden, der in den Haupttext eingefügt wurde.

MODUL 1.3.1

```
11000 REM#*******************************
11010 REM INITIALISIERUNG
12000 REM#*******************************
12010 PRINT"♥";:DIM TEXT$(500):LL=1:PL=1
12020 TEXT$(0)="[GELB]▼▼▼▼▼▼▼▼▼▼▼
▼▼▼▼▼▼▼▼▼▼▼▼▼"
12030 TEXT$(1)="[GELB][REVERS EIN]◤▼▼◥
◤▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼◥[REVER
S AUS]"
```

```
12040 DEF FNA(P)=1024+20*40+P
12050 DEF FNB(P2)=1024+40*P2
12060 GOSUB 14110
```

Mit diesem Modul werden die Hauptvariablen initialisiert und die Textanfangs- und Textendemarken in die Hauptmatrix gesetzt.

MODUL 1.3.2

```
13000 REM#******************************
13010 REM ZEILE EDITIEREN
13020 REM*******************************
13030 A$=" "
13040 P=0:PRINT "[HELLBLAU]SQQQQQQQQQQQQ
";a$
13050 CH=PEEK(FNA(P)):POKE 54272+FNA(P),
14:POKE FNA(P),160
13060 FOR TT=1 TO 5:NEXT TT:POKE FNA(P),
CH
13070 GET T$:IF T$="" THEN GOTO 13050
13080 IF T$=CHR$(13) OR LEN(A$)=81 THEN
GOSUB 14000:GOTO 13050
13090 IF T$="↑" THEN GOSUB 15000:POKE FN
B(PL-SS),62:GOTO 13040
13100 IF T$="←" AND P<>0 THEN 13040
13110 IF T$="←" ANDP=0 THEN P=LEN(A$)-1:
GOTO 13150
13120 IF P>0 AND T$=CHR$(20) THEN A$=LEF
T$(A$,P-1)+MID$(A$,P+1):P=P-1:GOTO13150
13130 IF T$=CHR$(20) THEN 13050
13140 IF T$<>""ANDT$<>"█" AND T$<>"█" TH
EN A$=LEFT$(A$,P)+T$+MID$(A$,P+1):P=P+1
13150 PRINT "SQQQQQQQQQQQQQQQQQQQ";A$:I
F T$="█" AND P>0 THEN P=P-1
13160 IF T$="█" AND P<LEN(A$)-1 THEN P=P
+1
13170 GOTO 13050
```

Mit diesem Modul kann der Benutzer bis zu zwei Bildschirm-Textzeilen am Ende des Bildschirms eingeben und bearbeiten, und die sich in Vorbereitung befindli-

chen Zeilen bearbeiten, um sie mit einem nachfolgenden Modul in den Haupttext einzufügen.

Kommentar

Zeilen 13050-13070: Hier stoßen wir das erste Mal auf eine Routine für den blinkenden Cursor. Auf der Grundlage der vom Benutzer definierten Funktion A überprüfen diese Zeilen den Bildschirmspeicher an der Stelle, die von der Variablen P angegeben wird. Sie erhalten den Code des dort stehenden Zeichens. Danach wird ein inverses blaues Leerzeichen mit POKE in dieselbe Adresse gesetzt, bleibt dort während der Dauer einer kurzen Schleife und wird dann durch das Originalzeichen ersetzt. Fand keine Eingabe über die Tastatur statt, wie durch die GET-Anweisung angegeben, so wird das Verfahren wiederholt.

Zeile 13080: Durch Betätigung der RETURN-Taste wird die Zeile in den Haupttext eingefügt - ein weiteres Modul wird benötigt. Die Zeile wird auch automatisch eingefügt, wenn die Länge zwei Bildschirmzeilen überschreitet - dadurch werden die letzten Zeichen unter Umständen aus der Zeile gelöscht.

Zeile 13090: Durch Betätigung des Symbols '↑' kann der Benutzer in das nachfolgende Modul gehen, das sich auf den Haupttext bezieht.

Zeilen 13100-13110: Durch Betätigung des linken Pfeilsymbols oben links auf der Tastatur wird der Cursor entweder an den Anfang oder das Ende der Zeile bewegt, je nachdem, an welcher Stelle er gerade steht.

Zeilen 13120-13130: Vorausgesetzt, der Cursor steht
nicht am Anfang der Zeile, so wird durch Betätigung der DELETE-Taste das Zeichen vor dem Cursor gelöscht. Wird dies berücksichtigt, so bleiben bei Benutzung von GET die Steuertasten, wie beispielsweise DELETE, wirkungslos, wenn nicht ein PRINT-Befehl benutzt wird. Wird also nicht gedruckt, so kann ihre Funktion neu definiert werden.

Zeile 13140: Handelt es sich bei dem eingegebenen Zeichen nicht um ein Zeichen für die Cursorbewegung, so wird davon ausgegangen, daß es sich um ein auszudruckendes Zeichen handelt. Es wird zu der Zeichenfolge an der Cursorposition hinzugefügt. Steht der Cursor in der Mitte der Zeichenfolge, so wird das Zeichen an dieser Stelle eingefügt - es ersetzt das Zeichen unter dem Cursor nicht.

Zeilen 13150-13160: Die Zeichenfolge wird in ihrer geänderten Form neu ausgedruckt. Handelte es sich bei dem eingegebenen Zeichen um ein Cursorzeichen, so wird die Cursorposition dementsprechend geändert.

Testen von Modul 1.3.2

Wird bei Zeile 14000 eine vorübergehende Zeilenschaltung eingefügt, so muß es nun möglich sein, Text am Ende des Bildschirms einzugeben und diesen Text zu bearbeiten.

MODUL 1.3.3

```
14000 REM#******************************
14010 REM ZEILE EINFUEGEN
14020 REM*******************************
14030 X=0
14040 IF LEN(A$)<41 THEN TT$(X)=LEFT$(A$
,LEN(A$)-1):A$="":GOTO 14070
14050 FOR I=41 TO 1 STEP-1:IF MID$(A$,I,
1)<>" " THEN NEXT I:I=41
14060 TT$(X)=LEFT$(A$,I-1):A$=MID$(A$,I+
1)
14070 X=X+1:IF A$<>"" AND A$<>" " THEN G
OTO 14040
14080 FOR I=LL+X TO PL+X STEP-1:TEXT$(I)
=TEXT$(I-X):NEXT I
14090 FOR I=0 TO X-1:TEXT$(PL+I)=TT$(I):
NEXT
14100 A$=" ":P=0:PRINT "■";:LL=LL+X:PL=P
L+X
14110 SS=PL-7:IF LL-PL<8 THEN SS=LL-15
14120 PRINT "■[HELLBLAU][illegible]
";a$;"";:if ss<0 then ss=0
14130 FOR I=SS TO SS+15:PRINT "[GELB]";T
EXT$(I);:IF LEN(TEXT$(I))<40 THEN PRINT
14140 IF I=PL-1 THEN PRINT CHR$(62)
14150 NEXT I:PRINT "[REVERS EIN][GRUEN]
                                        "
:RETURN
```

Mit diesem Modul wird die laufende Zeile in den Haupttext eingegeben.

Kommentar

Zeile 14040: Umfaßt die eingefügte Zeile weniger als 41 Zeichen, so wird sie in die von dem Bearbeitungs-Cursor > angegebene Position gesetzt.

Zeilen 14050-14070: Ist die Zeile länger, so wird mit diesen Zeilen nach dem Ende des letzten Wortes gesucht, das voll auf die Zeile paßt. Sämtliche links davon stehenden Zeichen werden zur ersten einzufügenden Zeile und werden in TT$ gespeichert. A$ wird nun als Rost definiert. Das Verfahren wird dann wiederholt. Mit der Variablen X wird angegeben, wie viele Zeilen sich ergeben.

Zeilen 14080-14090: Der Haupttext wird ab dem Bearbeitungs-Cursor verschoben, um Platz für die X neuen Zeilen zu schaffen. Danach werden die neuen Zeilen eingefügt.

Zeilen 14100-14110: A$ wird auf ein einziges Leerzeichen festgelegt, die Position des blinkenden Cursors wird auf Null festgelegt und der Bearbeitungs-Cursor wird durch die neuen Zeilen nach unten bewegt. Der Beginn der Haupttext-Anzeige wird neu definiert, so daß der Bearbeitungs-Cursor etwa in der Mitte stehen bleibt.

Testen von Modul 1.3.3

Nun sollte es möglich sein, Textzeilen durch Betätigung der RETURN-Taste in den Haupttext einzufügen.

MODUL 1.3.4

```
15000 REM#*****************************
15010 REM EINSETZEN DER ZEILE
15020 REM******************************
15030 P2=PL-SS
15040 GET T1$:IF T1$<>"" THEN 15070
15050 POKE 54272+FNB(P2),8:POKE FNB(P2),
62:FORIT I=1 TO 20:NEXT
15060 POKE FNB(P2),32:GOTO 15040
15070 PL=PL+(T1$="□")+10*(T1$="U"):IF PL
<1 THEN PL=1
15080 PL=PL-(T1$="■")-10*(T1$="D"):IF PL
>LL THEN PL=LL
15090 IF T1$=CHR$(13) THEN RETURN
15100 IF PL=>LL OR T1$<>CHR$(20) THEN 15
120
15110 LL=LL-1:FOR I=PL TO LL:TEXT$(I)=TE
XT$(I+1):NEXT:TEXT$(LL+1)=""
15120 IF PL<LL AND T1$="C" THEN A$=TEXT$
```

```
(PL)+"  ":RETURN
15130 IF T1$="P" THEN GOSUB 17000
15140 IF T1$="S" THEN GOSUB 18000
15150 IF T1$="F" THEN GOSUB 16000
15160 GOSUB 14110:GOTO 15030
```

Mit diesem Modul kann der Hauptbearbeitungs-Zeiger in dem Haupttext hin- und herbewegt werden, so daß Zeilen an verschiedenen Stellen eingefügt werden können. Mit diesem Modul kann der Benutzer auch andere Moduln aufrufen, die den Text formatieren, den Text an einen Drucker ausgeben oder auf Band sichern.

Kommentar

Zeilen 15040-15060: Routine des blinkenden Cursors für den Hauptbearbeitungs-Cursor.

Zeilen 15070-15080: Mit diesen beiden Zeilen wird der Hauptbearbeitungs-Cursor nach oben und unten bewegt. Verschiebungen um eine einzelne Zeile werden mit den normalen Cursortasten ausgeführt. Wird U oder D betätigt, so werden zehn Zeilen übersprungen. Hier wird auf die Benutzung logischer Bedingungen zur Ausführung dieser Verschiebungen verwiesen. Der Ausdruck (T1$'U') verfügt über einen Wert Null, wenn die Bedingung falsch ist und über einen Wert von minus Eins, wenn sie wahr ist. Mit ihm kann also eine IF-Anweisung, wie beispielsweise IF T1$'U'THEN usw., auf wirtschaftliche Weise ersetzt werden.

Zeile 15090: Durch Betätigung der RETURN-Taste wird zu dem Texteingabe-Modul zurückgegangen.

Zeilen 15100-15110: Durch Betätigung der DELETE-Taste wird die Zeile unter dem Cursor gelöscht.

Zeile 15120: Durch Betätigung von C wird die Zeile unter dem Cursor zur weiteren Bearbeitung an das Ende des Bildschirms kopiert.

Testen von Modul 1.3.4

Nun sollte es möglich sein, den Hauptbearbeitungs-Cursor zu bewegen, Zeilen zu löschen und sie wieder an das Ende des Bildschirms zu kopieren.

MODUL 1.3.5

```
16000 REM#*******************************
16010 REM FORMATIERERN DER ZEILE
16020 REM*******************************
16030 FOR I=1 TO LL-2:IF TEXT$(I)="" OR
TEXT$(I+1)="" THEN 16120
16040 SP=40-LEN(TEXT$(I)):FOR J=1 TO LEN
(TEXT$(I+1))
16050 IF MID$(TEXT$(I+1),J,1)<>" " THEN
NEXT J:J=J-1
16060 IF SP<J OR J=LEN(TEXT$(I+1)) THEN
16090
16070 TEXT$(I)=TEXT$(I)+" "+LEFT$(TEXT$(
I+1),J-1)
16080 TEXT$(I+1)=MID$(TEXT$(I+1),J+1):GO
TO 16040
16090 IF LEN(TEXT$(I+1))=>SP THEN 16120
16100 TEXT$(I)=TEXT$(I)+" "+TEXT$(I+1)
16110 FOR J=I+1 TO LL:TEXT$(J)=TEXT$(J+1
):NEXT J:LL=LL-1:PL=PL-1:GOTO 16040
16120 NEXTI:RETURN
```

Mit diesem Modul wird der Text formatiert, d.h. neu angeordnet, so daß leere Stellen am Ende der Zeilen wenn möglich mit Wörtern aus den nachfolgenden Zeilen aufgefüllt werden.

Zeile 16030: Wird eine leere Zeile in den Haupttext eingegeben, so wird sie nicht formatiert. Mit leeren Zeilen können somit Absätze oder andere Zeilen voneinander getrennt werden, die der Benutzer nicht gemeinsam ausführen möchte.

Zeilen 16040-16080: Der Leerraum am Ende der Zeile wird berechnet. Außerdem wird geschätzt, ob am Anfang der nächsten Zeile ein Wort steht, das in den leeren Platz paßt - wenn ja, wird es übertragen.

Zeilen 16090-16110: Paßt die ganze nächste Zeile an das Ende der laufenden Zeile, so wird sie angefügt und die Datei neu angeordnet, um den sich ergebenden leeren Platz zu verwerten.

Testen von Modul 1.3.5

Wird eine Reihe von Zeilen mit einzelnen Wörtern in den Haupttext eingefügt, so

sollte der Benutzer nun in der Lage sein, in den Hauptbearbeitungs-Modus zu gehen, F zu betätigen und zu sehen, wie die Wörter zu fortlaufenden Zeilen werden. Es können auch kurze Zeilen in die Mitte des Haupttextes eingefügt und dann neu formatiert werden.

MODUL 1.3.6

```
17000 REM#******************************
17010 REM DRUCKERAUSGABE
17020 REM*******************************
17030 OPEN 1,4:X=1
17040 IF X=LL THEN 17100
17050 IF TEXT$(X)="" THEN PRINT#1,"":X=X
+1:GOTO 17040
17060 PRINT#1,TEXT$(X);"  ";
17070 IF X+1=LL THEN 17100
17080 PRINT#1,TEXT$(X+1):IF TEXT$(X+1)="
" THEN PRINT#1,""
17090 X=X+2:GOTO 17040
17100 PRINT#1,"":CLOSE1:RETURN
```

Mit diesem einfachen Modul wird die Kommunikation mit dem Drucker (Einheit 4) eröffnet und der Haupttext ausgedruckt. Der Text wird im 80-Spalten-Format gedruckt (d.h. zwei Bildschirmzeilen bilden eine gedruckte Zeile). Leerzeilen werden überall dort gedruckt, wo in dem Haupttext eine Leerzeile steht. Obwohl das Programm selbst problemlos mit Kleinbuchstaben fertig wird (gleichzeitige Betätigung der SHIFT- und COMMODORE-Tasten), erfordern die meisten Drucker einen besonderen Befehl, um tatsächlich Kleinbuchstaben auszugeben. Dieser Befehl wird hier nicht angegeben, da er von Drucker zu Drucker verschieden ist. In dem Drucker-Handbuch stehen die erforderlichen Informationen.

MODUL 1.3.7

```
18000 REM#******************************
18010 REM ABSPEICHERN DER DATEN
18020 REM*******************************
18030 PRINT "[illegible][WEISS]KASSETTE EINLEGEN,
 DANN [REVERS EIN]RETURN[REVERS AUS]--"
18040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
18050 PRINT "MOEGLICHE BEFEHLE:":PRINT "
```

```
[GRUEN]1)daten speichern"
18055 PRINT"2) DATEN LADEN"
18060 INPUT "▒[HELLROT]BITTE WAEHLEN SIE
:";Q:ON Q GOTO 18080,18120
18070 RETURN
18080 POKE 1,7:FOR I=1 TO 2000:NEXT
18090 OPEN 1,1,2,"TEXTED":PRINT#1,PL:PRI
NT#1,LL
18100 FOR I=0 TO LL:FF$=TEXT$(I)+"@":PRI
NT#1,FF$:NEXT I
18110 CLOSE1:RETURN
18120 OPEN 1,1,0,"TEXTED":INPUT#1,PL,LL
18130 FOR I=0 TO LL:INPUT#1,TEXT$(I):NEX
T:CLOSE1
18140 FOR I=0 TO LL
18150 IF TEXT$(I)<>"@" THEN TEXT$(I)=LEF
T$(TEXT$(I),LEN(TEXT$(I))-1)
18160 IF TEXT$(I)="@" THEN TEXT$(I)=""
18170 NEXT I:RETURN
```

Hier handelt es sich um das Standardmodul einer Datendatei.

Zusammenfassung

Die in diesem Programm benutzten Techniken zur Änderung von Angaben, während diese auf dem Bildschirm angezeigt werden, verdienen Aufmerksamkeit, da sie bei weitem die (für den Benutzer) einfachste Möglichkeit zur Änderung von Zeichenfolgen darstellen. Sie können in eine Vielzahl von Programmen geschrieben werden, in denen eingegebene Zeichenfolgendaten geändert werden müssen - gegebenenfalls einschließlich der meisten Programme in diesem Handbuch.

Texted: Zusammenfassung der Instruktionen

Texteingabemodus:

Textzeichen können bei der Position des blinkenden Cursors eingegeben werden.
Mit den Pfeiltasten nach links und rechts wird der Cursor über die Zeichenfolge bewegt.
'← ' bewegt den Cursor an den Anfang oder das Ende der Zeile. '↑ ' ruft das Hauptbearbeitungs-Modul auf.
Durch Betätigung der RETURN-Taste wird die aktuelle Zeichenfolge in den Haupttext gesetzt.

Hauptbearbeitungs-Modus:

RETURN	Rückkehr zum vorhergehenden Modus
U,D	Mit den Cursortasten nach oben und unten wird der Hauptbearbeitungs-Cursor bewegt
DELETE	Löscht die Zeile unter dem Hauptbearbeitungs-Cursor
C	Kopiert die Zeile unter dem Hauptbearbeitungs-Cursor
P	Sendet den Text an den Drucker
S	Sichert den Text auf Band
F	Formatiert den Text.

KAPITEL 2

PROGRAMMIERHILFEN

Nachdem nun einige der grundlegenden Funktionen des Commodore 64 vorgestellt wurden, sollen hier drei eng gepackte Programme vorgestellt werden. Sie stellen die wichtigsten Hilfsmittel dar, mit denen separate Programme gemischt, deren Zeilen neu numeriert und ganze Abschnitte problemlos gelöscht werden können. Diese Programme sind deshalb so eng gepackt, damit sie mit der Merge-Routine zusammengefügt und dann an das Ende bestehender Programme angefügt werden können, ohne allzuviel Speicherplatz zu belegen. Nachdem zusätzliche Abschnitte in das Programm gemischt wurden und das Programm neu numeriert wurde, kann sich die Delete-Routine selbst und die beiden anderen Programme problemlos löschen.

2.1 MERGE

Dieses Programm ist zusammen mit den beiden anderen in diesem kurzen Kapitel vorgestellten Programmen ein Muß für den Benutzer, der die modulare Programmierung ernst nimmt. Mit diesem kleinen Programm kann stundenlange Arbeit gespart werden, indem nützliche Moduln auf Band gespeichert und einfach durch Betätigung einer Taste auf dem Bandrecorder zusammengefügt werden. Bevor ich dieses Programm vorstelle, möchte ich Steve Beats von Commodore UK danken, der mir die Grundidee vermittelt hat, auf der dieses Programm beruht.
Das Programm enthält keine Moduln - bei acht Zeilen würde sich dies auch kaum lohnen. Und dennoch wird durch ein Programm wie dieses das Prinzip der modularen Programmierung verdeutlicht.
Das Programm nimmt einfach ein anderes Programm oder einen Programmabschnitt vom Band und gibt dieses bzw. diesen in den Commodore 64 ein, ohne daß die Gefahr besteht, die dort schon stehenden Daten zu verlieren - es sei denn, es werden identische Zeilennummern benutzt. In diesem Fall wird das erste Programm überschrieben.

MODUL 2.1.1

```
63990 PRINT"■";
63991 OPEN1,1,0,"TEST"
63992 POKE 184,96:POKE 186,1:POKE 152,1:
PRINT"■■"
```

```
63994 GET#1,A$:PRINT A$;:IF ST THEN 6399
9
63995 IF A$<>CHR$(13) THEN 63994
63996 PRINT"GOTO 63992S";:POKE 631,13:PO
KE 632,13:POKE 633,13
63997 POKE 198,3:END
63999 CLOSE1
```

Kommentar

Zeile 63991: Die Datei, die dieses Programm liest, besteht aus einer Auflistung eines anderen Programms, das mit dem Befehl OPEN 1,1,2'TEST':CMD1: LIST auf Band gespeichert wurde. Der CMD-Befehl bedeutet, daß jede Angabe, die normalerweise auf dem Bildschirm ausgegeben würde, in Wirklichkeit in die Datei mit der angegebenen Nummer gesendet wird - in diesem Fall eine auf dem Kassettenrecorder geöffnete Datei. Das einzig Besondere an dieser Datei ist die Sekundäradresse, d.h. 2. Mit ihr wird angegeben, daß es sich hier um eine Ausgabedatei handelt, an deren Ende eine besondere Dateiendemarke ausgedruckt wird. Die Auflistung des Programms wird demzufolge nicht an den Bildschirm gesendet, sondern an den Kassettenrecorder. Außerdem weist sie nicht dasselbe Format auf, unter dem ein Programm normalerweise gespeichert wird, sondern das ASCII-Format. Das Programm könnte auch Zeichen für Zeichen, wie auf dem Bildschirm dargestellt, aufgelistet werden. Stoppt der Kassettenrecorder, ohne daß er ausgeschaltet wird, so muß die Datei durch Eingabe von PRINT1: CLOSE1 beendet werden. Dadurch wird gewährleistet, daß die letzten Zeichen des Programms ausgedruckt werden und die Datei richtig abgeschlossen wird.

Zeilen 63994-63995: Während Zeile 63992 im Moment übersprungen wird, beginnt das Mischprogramm nun damit, die Zeichen des auf Band aufgelisteten Programms auszulesen, bis es auf einen RETURN-Code stößt, der das Ende einer Datei angibt.

Zeile 63996: Diese etwas seltsame Zeile ist in Wirklichkeit der Schlüssel zu dem Programm. Nachdem eine Zeile vom Band genommen und auf dem Bildschirm ausgedruckt wurde, druckt das Programm nun direkt unter der Zeile den Befehl GOTO 63992 sowie das Steuerzeichen für Cursor in Ausgangsposition aus. Im Anschluß daran werden drei RETURN-Codes mit POKE in den Tastaturpuffer gesetzt, den Speicherbereich, in dem Tasten gespeichert werden, die zwar betätigt, deren Funktion jedoch noch nicht ausgeführt wurde. Der dritte Code wird mit POKE in Adresse 198 gesetzt, in der festgehalten wird, wie viele Tastenfunktionen noch ausgeführt werden müssen. Nachdem dies geschehen ist, ist das Programm eigentlich beendet. Der Commodore 64 glaubt nun jedoch, daß die RE-

TURN-Taste drei Mal betätigt wurde und reagiert dementsprechend. Der Cursor wird also über die Zeile, die vom Band gelesen und auf dem Bildschirm ausgedruckt wurde, und über GOTO 63992 nach unten bewegt. Dadurch wird die Zeile in den Speicher eingegeben, genau so, als hätten Sie den Cursor über die Zeile gesetzt und die RETURN-Taste betätigt. Danach beginnt das Programm wieder bei 63992.

Zeile 63992: Diese Zeile mit den geheimnisvollen POKE-Befehlen löst ein grundlegendes Problem - wann immer eine neue Zeile in ein Programm eingegeben wird, werden sämtliche bestehenden Zeilen mit CLOSE abgeschlossen. Wird die erste Zeile des neuen Programms eingegeben, so wird die Datei auf dem Kassettenrecorder abgeschlossen. Sämtliche Instruktionen, Daten mit GET aus dieser Datei zu holen, führen zu einem Fehler. Die Datei kann nicht mit einem Basic-Befehl wieder geöffnet werden, da die Dateianfangsmarke schon überschritten wurde. Mit ihr wird dem Commodore 64 mitgeteilt, daß eine Datei auf dem Band gefunden wurde. Mit den POKE-Befehlen wird nun ein wenig geschummelt und dem C64 mitgeteilt, daß Datei Nr. 1 mit einer Sekundäradresse Null auf dem Kassettenrecorder geöffnet ist. Nun kann die zweite Zeile aus der Datei geholt werden usw. ad infinitum.

Zu irgendeinem Zeitpunkt stößt das Programm auf die Dateiendemarke. Mit ihr wird angegeben, daß die Auflistung beendet ist. Der zweite Teil von Zeile 63994 (IF ST THEN 63999) entdeckt dies und springt zur letzten Zeile. Normalerweise stoppt das Programm mit der Fehlermeldung OUT OF DATA. Dies bedeutet, daß der Mischvorgang erfolgreich ausgeführt wurde.
Im täglichen Gebrauch ist dieses Programm alles andere als schnell. Es wechselt zwischen einem leeren Bildschirm, auf den die Daten vom Band geladen werden, und blinkenden Zeilen am Anfang des Bildschirm. Für dieses Programm ist ein gutes Band erforderlich, da durch ein Band mit Ausfällen die ganze Sache erfolglos werden könnte. Bei etwas Vorsicht ist dies jedoch ein Programm, auf das von Zeit zu Zeit zurückgegriffen wird - ein Versuch ist durchaus der Mühe wert.

2.2 DELETE

Bei der Entwicklung von Programmen, bei denen ähnliche Moduln für vorher eingegebene Programme benutzt werden, ist es besonders nützlich, wenn das Originalprogramm geladen und nur die Teile gelöscht werden können, die für die neue Anwendung nicht benötigt werden. Die folgende, 12 Zeilen umfassende Routine, ermöglicht genau dies.
Die Routine beruht auf der äußerst einfachen und klaren Art und Weise, mit der

Programme im Speicher der Commodore Computer aufgelistet werden. Jede Programmzeile im Speicher beginnt mit zwei Verbindungsbytes, die die Startadresse der nächsten Zeile im Speicher angeben. Auf diese beiden Bytes folgen zwei weitere Bytes, in denen die tatsächliche Zeilennummer angegeben wird. Diese Routine durchsucht nun die Zeilennummern zwischen einer vom Benutzer angegebenen Anfangsnummer und einer ebenfalls vom Benutzer angegebenen Endnummer. Ist die Adresse der letzten zu löschenden Zeile gefunden, so setzt das Programm einfach die nächste Zeilenadresse in die erste zu löschende Zeile, um auf die Zeile hinter der letzten zu löschenden Zeile zu zeigen. Dadurch erhält man eine einzelne Zeile, die sich vom Anfang der ersten zu löschenden Zeile bis zum Ende der letzten zu löschenden Zeile erstreckt. Nun kann gelöscht werden, indem einfach die erste Zeile gelöscht wird - alle anderen Zeilen werden mitgelöscht.

MODUL 2.2.1

```
63700 INPUT "ERSTE ZU LOESCHENDE ZEILE :
";D1
63710 INPUT "LETZTE ZU LOESCHENDE ZEILE:
";D2
63715 DEF FNDH(X)=PEEK(X)+256*PEEK(X+1)
63716 DEF FNH1(X)=X AND 255
63717 DEF FNH2(X)=INT(X/256)
63720 LA=2049
63730 LN=FNDH(LA+2):IF LN<D1 THEN LA=FND
H(LA):GOTO 63730
63740 DSTART=LA
63750 LN=FNDH(LA+2):IF FNDH(LA)=0 THEN 6
3760
63755 IF LN<=D2 THEN LA=FNDH(LA):GOTO 63
750
63760 POKE DSTART,FNH1(LA):POKE DSTART+1
,FNH2(LA)
63770 POKE DSTART+4,143:FOR I=5 TO 10:PO
KE DSTART+I,33:NEXT
```

Kommentar

Zeile 63715: Diese Funktion, die in den verschiedensten Zusammenhängen äußerst nützlich sein kann, setzt eine aus zwei Bytes bestehende Zahl, mit der die meisten Computer allgemein arbeiten, in eine normale Dezimalzahl im Bereich

von 0 bis 65535 um. Die aus zwei Bytes bestehende Zahl verfügt in Wirklichkeit über eine Basis von 256, d.h. sie besteht aus maximal 255 Einheiten und einer zweiten Ziffer bis zu 255*256, so daß 99 aus neun Einheiten und 9 mal 10 besteht. Um jedoch etwas Verwirrung zu schaffen, werden die Ziffern rückwärts gespeichert, d.h. das höherwertige Byte steht an zweiter Stelle.

Zeilen 63716-63717: Diese beiden vom Benutzer definierten Funktionen führen genau die entgegengesetzte Aufgabe durch, d.h. setzen eine Dezimalzahl in eine aus zwei Bytes bestehende Zahl um.

Zeile 63720: LA entspricht der Anfangsadresse der ersten Programmzeile.

Zeile 63730: LN entspricht dem Wert des dritten und vierten Bytes der Zeile - der Zeilennummer. Ist dieser Wert kleiner als der Wert der ersten zu löschenden Zeile, so benutzt LA die beiden Verbindungsbytes, um zu der Anfangsadresse der nächsten Zeile zu springen. Dieses Verfahren wird wiederholt, bis eine Zeilennummer gefunden wird, die gleich oder größer als die erste zu löschende Zeilennummer ist.

Zeile 63740: Die Anfangsadresse der ersten zu löschenden Zeile wird in der Variablen DSTART gespeichert.

Zeile 63750: Mit der FNDH Funktion überprüft die Variable LA den Speicher von Zeilenanfang zu Zeilenanfang. Mit jedem Sprung wird die Variable LN der dort gefundenen Zeilennummer gleichgesetzt. Findet FNDH eine Speicheradresse mit einer Null an einer Stelle, an der normalerweise ein Paar mit Verknüpfungsbytes stehen sollte, so ist das Ende des Programms erreicht.

Zeile 63755: Jedesmal, wenn eine neue Zeilennummer angetroffen wird, wird sie mit der Nummer der letzten zu löschenden Zeile verglichen. Ist die letzte Zeile noch nicht erreicht, so wird der nächste Sprung vorgenommen.

Zeile 63760: Hat das Programm diesem Punkt erreicht, so wurde die letzte zu löschende Zeile gefunden. In die beiden Verbindungsbytes bei DSTART wird die Adresse der Zeile hinter der letzten zu löschenden Zeile gesetzt.

Zeile 63770: Das erste Zeichen des neuen einzelnen zu löschenden Zeilenblocks wird in eine REM-Anweisung umgesetzt, und hinter REM wird eine Folge von Ausrufezeichen gesetzt, um die zu löschende Zeile zu markieren.
Die Routine hat nun ihre Funktion erfüllt. Jetzt braucht nur noch die Nummer der markierten Zeile eingegeben und die RETURN-Taste betätigt zu werden - der ganze Block, von einigen wenigen Zeilen bis zu einem vollständigen Programm, wird gelöscht.

In der Praxis wird diese Routine am besten mit der Merge-Routine im vorhergehenden Abschnitt dieses Kapitels benutzt (die Merge-Routine wird geladen und diese Routine wird entweder hinzugefügt oder eingemischt), da die Routine auf diese Weise zu bestehenden Programmen hinzugefügt werden kann, aus denen einige Zeilen benutzt und andere gelöscht werden sollen. Diese Routine ist in wenigen Augenblicken ausgeführt und spart umfangreiche Tastaturarbeiten

2.3 RENUMBER

Eigentlich ist es der Wunsch eines jeden Benutzers, mit sauber numerierten Programmen arbeiten zu können - der Unterschied zwischen einen professionell aufgebauten Programm und einem von vornherein schlampig aussehenden Programm ist gewaltig. Mit dem relativ kurzen, hier vorgestellten Programm kann jedes Programm neu numeriert werden, obwohl es nicht gerade als schnell zu bezeichnen ist und dem Bereich der Programmzeilen eine einfache Einschränkung auferlegt.
Das Programm numeriert jedes beliebige Programm neu, einschließlich der GOTOs, GOSUBs, ONs...GOTOs und GOSUBs und der auf IF...THEN folgenden Zeilennummern. Allerdings verschiebt es das Programm im Speicher nicht, so daß keine Ziffern zu einem GOTO (usw.) addiert oder subtrahiert werden können. Hierzu müßte das ganze Programm verschoben werden, das auf die geänderte Adresse folgt. Nicht daß dies unmöglich oder auch besonders schwierig wäre, um diese Funktionen jedoch einigermaßen praktisch zu gestalten, müßen sie im Maschinencode ausgeführt werden, der nicht in den Rahmen dieses Handbuchs fällt.
Aufgrund dieser Einschränkung beginnen sämtliche Programme in diesem Handbuch, die mit Hilfe der nachfolgend beschriebenen Routine neu numeriert wurden, bei 11000. (Unregelmäßige Zeilennummern sind auf Änderungen zurückzuführen, die erst im späteren Verlauf dieses Verfahrens vorgenommen wurden.) Auf diese Weise wird gewährleistet, daß sämtliche Zeilennummern über fünf Ziffern verfügen.
Die Art und Weise, in der das neu numerierte Programm strukturiert ist, kann durch Benutzung von Formatierzeilen innerhalb des neu zu numerierenden Programms gesteuert werden. Betrachtet man die Programme in diesem Handbuch näher, so wird man feststellen, daß die Moduln fast ausnahmslos mit einer REM-Anweisung beginnen, und daß das erste Zeichen hinter REM ein " Symbol ist. Das Programm zur Neunumerierung ist so beschaffen, daß es mit der Neunumerierung bei 11000 beginnt und in Zehnerschritten erhöht, bis es zu einer Zeile 11000 kommt. Diese Zeilennummer wird dann um 1000 erhöht.
Dies führt nicht nur zu gut leserlichen Programmen, sondern bedeutet, daß die

Struktur des neu zu numerierenden Programms kontrolliert werden kann. Angenommen, Sie verfügen über ein Programm, aus dem drei oder vier Moduln benutzt werden sollen, die aktuellen Zeilennummern stimmen jedoch nicht mit der Struktur überein, die für das neue Programm gewünscht wird. Beispielsweise ist zwischen den einzelnen Moduln nicht ausreichend Platz vorhanden, um einen anderen, auf Band stehenden Abschnitt einmischen zu können. Durch Einfügung von zwei REM-Anweisungen mit einem ersten Zeichen von " und nachfolgende Neunumerierung wird automatisch eine Lücke von 2000 geschaffen, bei der die REM-Anweisungen stehen - nicht allzu schwierig

MODUL 2.3.1

```
63000 CLR:DIM ZZ(500,1):LA=2049:PP=LA
63010 DEF FNDH(X)=PEEK(X)+256*PEEK(X+1)
63013 DEF FNH1(X)=X AND 255
63016 DEF FNH2(X)=INT(X/256)
63050 IF PP<>FNDH(LA) THEN 63060
63053 LA=FNDH(LA)::NL=FNDH(LA+2):IF NL=6
3000 THEN GOTO 63500
63059 PP=PP+4
63060 IF PEEK (PP)<>167 THEN 63070
63062 S=0:IF PEEK(PP+1)=32 THEN S=1
63064 IF PEEK (PP+1+S)<48 OR PEEK (PP+1+
S)>57 THEN 63200
63065 GOTO 63076
63070 IF PEEK (PP)<>137 AND PEEK(PP)<>14
1 THEN 63200
63076 LET S=0:IF PEEK(PP+1)=32 THEN S=1
63080 GG$="":FOR I=1+S TO 5+S:IF PEEK(PP
+I)<48 OR PEEK(PP+I)>57 THEN GOTO 63140
63085 LET GG$=GG$+CHR$(PEEK(PP+I)):NEXT
63090 GG=VAL(GG$):L1=2049:L2=FNDH(2051):
LL=11000
63093 IF L2=63000 THEN PRINT "SPRUNG ZU
NICHT EXISTENTER ZEILE IN ZEILE";NL:STOP
63095 IF L2=GG THEN 63100
63097 L1=FNDH(L1):L2=FNDH(L1+2):LL=LL+10
63098 IF PEEK(L1+4)=143 AND PEEK(L1+5)=3
5 THEN LL=1000*INT((LL+1000)/1000)
63099 GOTO 63093
63100 LET ZZ(ZI,0)=LL:LET ZZ(ZI,1)=PP+S:
```

```
ZI=ZI+1
63110 IF PEEK(PP+S+6)=44 THEN PP=PP+6+S:
GOTO 63076
63135 GOTO 63200
63140 PRINT "UNGUELTIGER BEFEHL IN ZEILE
";NL:STOP
63200 PP=PP+1:GOTO 63050
63500 LA=2049:LL=10000
63503 IF PEEK(LA+4)=143 AND PEEK(LA+5)=3
5 THEN LL=1000*INT((LL+1000)/1000)
63505 IF FNDH(LA+2)=63000 THEN 63600
63510 POKE LA+2,FNH1(LL):POKE LA+3,FNH2(
LL)
63520 LET LL=LL+10:LET LA=FNDH(LA):GOTO
63503
63600 IF ZI=0 THEN STOP
63605 FOR I=0 TO ZI-1
63610 FOR J=1 TO 5:POKE ZZ(I,1)+J,ASC(MI
D$(STR$(ZZ(I,0)),J+1)):NEXT J
63620 NEXT I
63650 STOP
```

Kommentar

Zeile 63000: Im Feld ZZ werden die Adressen der neu zu numerierenden GOTO-Anweisungen usw., sowie die neu zuzuweisende Nummer aufgezeichnet.

Zeilen 63010-63016: Dieselben Funktionen wie in der Routine für das Löschen von Blöcken. Nachdem die beiden gemischt wurden, muß diese Gruppe gelöscht werden.

Zeilen 63050-63509: PP ist ein Zeiger, der den Speicher nach GOTO usw. absucht. Er beginnt am Anfang des Programmbereichs bei 2049. Jedesmal, wenn der Anfang einer neuen Zeile erreicht wird, wird die Zeilenadreß-Variable (LA) erhöht. Die Zeilennummer der aktuellen Zeile wird in LN gespeichert. Das Programm stoppt bei Zeile 63000 - dem Beginn dieser Routine. Nun springt PP zu dem ersten Zeichen der Zeile.

Zeilen 63060-63065: Mit diesen Zeilen wird bei Antreffen einer THEN-Anweisung im Speicher geprüft, ob auf THEN entweder direkt oder im Anschluß an ein Leerzeichen eine Zahl folgt. Mit der Variablen S wird einfach aufgezeichnet, ob ein

Leerzeichen vorhanden ist. Folgt auf THEN keine Zahl, so geht PP weiter.

Zeilen 63070-63085: Wird der Code für GOSUB oder GOTO gefunden, so prüft das Programm, ob ein Leerzeichen folgt oder nicht. GG$ wird aus den Ziffern der Zeilenbestimmung erstellt. Weniger als fünf Ziffern führen zu einer Fehlermeldung in Zeile 63140.

Zeilen 63090-63099: GG wird gleich der GOTO- oder GOSUB-Bestimmung gesetzt. Diese Routine durchsucht dann die Zeilennummern in dem Programm ab dem Anfang und sucht nach der Bestimmung. Für jede überprüfte Zeile wird die Variable LL um 10 erhöht, wobei bei 11000 begonnen wird. Auf diese Weise wird festgehalten, wie die Zeilennummer nach der Neunumerierung des Programms lautet - die Zeile kann hier nicht neu numeriert werden, da unter Umständen eine andere GOTO-Anweisung auf diese Zeile zeigt. Wird REM angetroffen, so wird LL auf den nächsthöheren 1000-er Wert erhöht.

Zeile 63100: An dieser Stelle wurde die richtige Zeilennummer gefunden, so daß die Adresse der GOTO-Anweisung im Feld ZZ zusammen mit ihrer künftigen Zeilennummer (LL) gespeichert wird.

Zeile 63110: Handelt es sich bei dem sechsten Zeichen hinter der GOTO- oder GOSUB-Anweisung um ein Komma, so wird davon ausgegangen, daß es sich hier um eine ON...GOTO-GOSUB-Anweisung handelt. PP wird weiter bewegt und die neue Zeilenbestimmung wird von einem früheren Teil der Routine aufgenommen.

Zeile 63200: Das Verfahren wird fortgesetzt, während PP weiter durch den Speicher geht, bis Zeile 63000 angetroffen wird.

Zeilen 63500-63520: Diese Zeilen beginnen am Anfang des Programms und numerieren nur die Zeilen neu (wobei REM nicht vergessen werden darf).

Zeilen 63600-63650: Nun brauchen nur noch die Adressen sämtlicher GOTO-GOSUB-Bestimmungen aus ZZ herausgenommen und die neuen Bestimmungen mit POKE in die fünf auf jede Adresse folgenden Bytes gesetzt werden. Sie wurden schon berechnet.

Obwohl dieses Programm, was Geschwindigkeit oder Flexibilität betrifft, nicht mit einem guten Dienstprogramm im Maschinencode verglichen werden kann, führt es doch seine Funktion aus, wie sich aus den Programmen in diesem Handbuch ergibt. Verfügen Sie nicht über eine Neunumerierungs-Routine im Maschinencode, so bin ich sicher, daß Sie auf diese Routine häufiger zurückzugreifen werden, als auf nahezu alle anderen.

Wird diese Routine mit den beiden vorhergehenden Dienstprogrammen gemischt, wobei die drei jeweils durch STOP-Anweisungen voneinander getrennt werden, so verfügen Sie über ein leistungsfähges Hilfsmittel, das die Programmierung einfacher und die Programme leserlicher gestaltet.

KAPITEL 3

DER FARBIGE C64

Der Commodore 64 verfügt über ein faszinierendes Spektrum an Grafik-Möglichkeiten. Die Formen und Farben, die auf dem Bildschirm angezeigt werden können, decken nahezu jeden nur erdenklichen Bedarf ab und reichen sicherlich aus, einen Hobbykünstler für den Rest seines Lebens zu beschäftigen. In diesem Kapitel werden vier Grafik-Programme vorgestellt, mit denen Grafik-Zeichenvorräte, vom Benutzer definierte Zeichen, Sprites und Bit-Raster-Grafiken entdeckt werden können. Die hier vorgestellten vier Programme stellen sicherlich nur einen kleinen Teil dessen dar, was der C64 vollbringen kann. Deshalb sind die Programme als Hilfsmittel gedacht, die es Ihnen ermöglichen sollen, all die farbigen Funktionen in Ihre eigenen Programme einzufügen, die diese dann aus der Masse der anderen Programme hervorheben.

3.1 ARTIST

Nur wenige Homecomputer verfügen über einen so leistungsfähigen Grafik-Zeichenvorrat wie die Commodore-Systeme. Mit den über die Tastatur einzugebenden Zeichenkombinationen gibt es eigentlich nichts, was nicht in irgendeine grafische Form gebracht werden könnte. Dies ist besonders nützlich, wenn die Ausgabe der meisten Programme belebt werden soll, insbesondere in Verbindung mit den ausgezeichneten Farbmöglichkeiten des C64.
Die einzige Einschränkung liegt im kreativen Bereich bei der Entwicklung der Grafik-Anzeigen. Dies kann natürlich ausschließlich mit Druck-Anweisungen in einem Programm erfolgen. Die Druck-Anweisungen jedoch mit einer Vielzahl von Farb-Befehlen, Befehlen für die inverse Anzeige usw. in genau die richtige Form zu bringen, wobei jede Zeile separat definiert werden muß, kann sehr zeitaufwendig sein. Hier wird einfach eine Möglichkeit benötigt, den Bildschirm wie eine Staffelei zu benutzen, wobei mit Grafikzeichen in den verschiedensten Farben gezeichnet, radiert und beliebig geändert werden kann. Schließlich muß eine Möglichkeit bestehen, den Entwurf für die Nachwelt oder zumindest für andere Programme, die den erstellten Entwurf benutzten könnten, auf Band zu sichern. All dies wird mit dem jetzt vorgestellten Programm erreicht.
Im Laufe dieses Verfahrens werden auch viele Informationen über die Handhabung des Bildschirm- und Farbspeichers, zusammen mit nützlichen Speicheradressen für die Kontrolle von Eigenschaften, wie beispielsweise der Druckfarbe, vermittelt.

Artist: Variablentabelle

CC	Aktuelle Cursorposition
CO(3)	Koordinaten von zwei Ecken des zu sichernden Entwurfs
CT	Vorübergehender Speicherplatz der Cursorposition
CU	Aktuelle Cursor-Farbe
C1$	Werte der zu sichernden Zeichen in dem Entwurf
D2$	Farbwerte der zu sichernden Zeichen
D3$,D4$	Temporäre Kopien von D1$ und D2$
MODE	Definiert, welche der vielen Farb-Eigenschaften adressiert wird
PC	Farbe des Zeichens in Position PP
PP	Ursprünglicher Inhalt der aktuellen Cursorposition
PT	Speicheradresse der beiden in Feld CO festgehaltenen Ecken

MODUL 3.1.1

```
11000 REM#*******************************
11010 REM VARIABLEN
11020 REM*******************************
11030 R$=CHR$(13)
```

Eigentlich kann man dies kaum als Modul bezeichnen. Mit ihm kann jedoch ein entsprechender Bereich für alle erforderlichen Variablen reserviert werden, sofern Sie sich entscheiden, das Programm weiter zu entwickeln. Bei der tatsächlich definierten Zeichenfolge handelt es sich um ein Standard-Trennzeichen für eine Datendatei.

MODUL 3.1.2

```
12000 REM#*******************************
12010 REM CURSORBEWEGUNG/DARSTELLUNG
12020 REM*******************************
12030 PRINT"■";
12040 POKE 650,255:GET A$
12050 CC=PEEK(211)+PEEK(210)*256+PEEK(20
9):PP=PEEK(CC):PC=PEEK(CC+54272)
12060 POKE CC,42:POKE CC+54272,CU:FOR I=
1TO15:NEXT:POKE CC,PP:POKE CC+54272,PC
12070 IF A$="" THEN 12040
12080 IF CC>1983 AND A$="■" THEN 12040
12090 IF CC=2023 AND A$="■" THEN 12040
12100 IF CC=1024 AND A$="■" THEN 12040
```

```
12110 IF CC<1064 AND A$="□" THEN 12040
12120 IF A$="□" OR A$="■" OR A$="■" OR A
$="■" THEN PRINT A$;:GOTO 12040
12130 IF A$=CHR$(133) THEN MODE=1
12140 IF A$=CHR$(137) THEN MODE=2
12150 IF A$=CHR$(134) THEN MODE=3
12155 IF A$=CHR$(138) THEN 12030
12160 IF (MODE=1 OR MODE=2 OR MODE=3) AN
D A$="←" THEN MODE=MODE+.5:GOTO 12040
12170 IF A$=CHR$(135) THEN INV=(INV=0)
12180 IF A$=CHR$(139) THEN MODE=6
12190 IF A$=CHR$(136) THEN MODE=7
12200 IF A$=CHR$(140) THEN MODE=8
12210 IF MODE=1 AND A$>="1" AND A$<="8"
THEN POKE CC+54272,VAL(A$)-1:GOTO 12040
12220 IFMODE=1.5ANDA$>="1" AND A$<="8" T
HEN POKE CC+54272,8+VAL(A$)-1:GOTO 12040

12230 IF MODE=2 AND A$>="1" AND A$<="8"
THEN POKE 53281,VAL(A$)-1:GOTO 12040
12240 IF MODE=2.5 AND A$>="1" AND A$<="8
" THEN POKE 53281,8+VAL(A$)-1:GOTO 12040

12250 IF MODE=3 AND A$>="1" AND A$<="8"
THEN POKE 646,VAL(A$)-1:GOTO 12040
12260 IF MODE=3.5 AND A$>="1" AND A$<="8
" THEN POKE 646,8+VAL(A$)-1:GOTO 12040
12270 IF MODE<>6 OR (A$<>"R" AND A$<>"D"
 AND A$<>"S") THEN 12320
12280 IF A$="R" THEN GOSUB 13000
12290 IF A$="D" THEN GOSUB 13060
12300 IF A$="S" THEN GOSUB 14000
12310 CC=CT:GOTO 12040
12320 IF MODE=8 AND A$>="1" AND A$<="8"
THEN CU=VAL(A$)-1:GOTO 12040
12330 IF INV=-1 THEN PRINT "[REVERS EIN]
";
12340 PRINT A$;"[REVERS AUS]";
12350 GOTO 12040
```

Dieses Modul reicht schon aus, um den Bildschirm in eine Staffelei zu verwandeln. Mit ihm können Sie einen blinkenden Cursor auf dem Bildschirm hin- und herbewegen, Zeichen ausdrucken, ändern und löschen, die Vordergrund- und Hintergrundfarbe für Zeichen ändern ...

Kommentar

Zeilen 12040-12070: Diese Routine erzeugt einen blinkenden Cursor, der vom Benutzer gesteuert wird.

Zeile 12040: Mit diesem POKE-Befehl wird die Wiederholfunktion festgelegt, so daß eine Taste, die gedrückt gehalten wird, ständig dasselbe Zeichen ausdruckt. Mit dem zweiten Teil der Zeile wird ein einzelnes über die Tastatur eingegebenes Zeichen empfangen.

Zeile 12050: CC ist gleich der Speicheradresse der aktuellen Druckposition. PEEK(211) gibt die Position entlang der Zeile (0-39) an, während PEEK(210)*256+PEEK(209) die Speicheradresse des Zeilenanfangs angibt. PP wird gleich dem Bildschirmcode des Zeichens gesetzt, das gerade die Position belegt, an der der Cursor aufblinkt. PC entspricht der Farbe des Zeichens in dieser Position.

Zeile 12060: Ein Sternchen, Bildschirmcode 42, wird nun mit POKE in die Position gesetzt, an der der Cursor blinken soll. Die aktuelle Cursorfarbe CU wird mit POKE an der entsprechenden Stelle in den Farbspeicher gesetzt. Dank einer kurzen Zeitschleife bleibt das Sternchen einen Augenblick lang auf dem Bildschirm stehen. Danach werden das Originalzeichen (Code PP) und die Originalfarbe (PC) wieder mit POKE in den Speicher gesetzt.

Zeile 12070: Wurde keine Taste betätigt, so wird der Zyklus wiederholt.

Zeilen 12080-12110: Mit diesen Zeilen wird geprüft, ob der Cursor bei Betätigung der Cursortasten nicht vom Bildschirm läuft.

Zeile 12120: Wird ein Cursor-Steuerzeichen eingegeben und besteht es die Tests in den vier oben angegebenen Zeilen, so wird es sofort ausgedruckt. Das Programm kehrt dann zu der Routine für den blinkenden Cursor zurück.

Zeilen 12130-12200: Werden die Funktionstasten auf der rechten Seite der Tastatur als Eingabetasten benutzt, so kann der Benutzer mit diesen Zeilen die verschiedenen Betriebsarten angeben, mit denen verschiedene Farbeigenschaften festgelegt werden können.

Zeile 12130: Wird die Taste f1 betätigt, so wird das Programm in den MODUS 1 versetzt. Wird in diesem Modus eine der Tasten 1 bis 8 betätigt, so wird die Farbe jedes Zeichens neu definiert, über dem der Cursor gerade steht. Als Farbe wird die Farbe benutzt, die auf der Vorderseite der Taste angegeben wird.

Zeile 12140: Bei Betätigung von f2 kann mit demselben Verfahren die Hintergrundfarbe des Bildschirms neu definiert werden.

Zeile 12150: Durch Betätigung von f3 kann die Druckfarbe mit demselben Verfahren zurückgesetzt werden.

Zeile 12160: Da in Wirklichkeit 16 Farben zur Verfügung stehen, wird durch Betätigung der Pfeil-nach-links-Taste, oben links auf der Tastatur, im MODUS 1, 2 oder 3 der Modus neu definiert. So wird durch Betätigung der Tasten 1 bis 8 die Farbe gewählt, die normalerweise durch Betätigung dieser Taste zusammen mit der Commodore-Taste erreicht wird. Die neu definierten Eigenschaften entsprechen den Eigenschaften, auf die im Hauptmodus Bezug genommen wird.

Zeile 12170: Bei Betätigung von f5 wird die inverse Anzeige gesetzt oder zurückgesetzt - so daß Zeichen invers ausgedruckt werden können.

Zeile 12180: Bei Betätigung von f6 kann der erstellte Entwurf gesichert werden. Das richtige Verfahren wird später im einzelnen erläutert.

Zeile 12190: Durch Betätigung von f7 wird nur die Neudefinition in einen nicht wirksamen Modus vorgenommen. Dadurch kann der Benutzer die Zahlen 1 bis 8 auf dem Bildschirm ausdrucken, anstatt eine Farbeigenschaft neu zu definieren.

Zeile 12200: Durch Betätigung von f8 kann der Benutzer die Cursorfarbe in eine der ersten acht Farben ändern. Dies ist besonders nützlich, wenn die Bildschirmfarbe so neu definiert wurde, daß der Cursor nicht mehr deutlich sichtbar ist.

Zeilen 12210-12220: Bei dem MODUS 1 oder 1,5 wird die Farbeingabe für das aktuelle Quadrat mit POKE in den Farbspeicher gesetzt.

Zeilen 12230-12240: Bei dem MODUS 2 oder 2,5 wird der neue Farbcode mit POKE in Adresse 53281 gesetzt. Mit dieser Adresse wird die Hintergrundfarbe des Bildschirms festgelegt.

Zeilen 12250-12260: Bei dem MODUS 3 oder 3,5 wird der neue Farbcode mit POKE in Adresse 646 gesetzt. Mit dieser Adresse wird die aktuelle Druckfarbe festgelegt.

Zeilen 12270-12310: Im MODUS 6 bezieht sich diese Routine auf die Sicherung kleiner oder großer Entwürfe. Wird R eingegeben, so kann die Definition eines Bildschirmrechtecks gesichert werden. Durch D wird ein Entwurf in kleinem Maßstab gesichert. Durch S wird der ganze Bildschirm auf Magnetband gesichert.

Zeile 12310: Mit CT wird die aktuelle Cursorposition gesichert, die dann während der SAVE-Routine geändert werden kann.

Zeile 12320: Im MODUS 8 wird der neue Farbcode in die Variable CU gespeichert.

Zeile 12330: Ist die inverse Anzeige gesetzt (INV-1), so wird das Steuerzeichen RVS ON ausgedruckt. Auf diese Weise wird das nächste auszudruckende Zeichen invers gedruckt.

Zeile 12340: Erreicht das Programm diesen Punkt, so wird jedes eingegebene Zeichen auf dem Bildschirm ausgedruckt und das Steuerzeichen für inverse Anzeige aus (RVS OFF) wird im Anschluß an dieses Zeichen ausgedruckt.

Testen von Modul 3.1.2

Nachdem dieses Modul eingegeben wurde, sollten Sie in der Lage sein, beliebige Entwürfe auf dem Bildschirm mit dem ganzen Zeichenvorrat zu erstellen, der über die Tastatur zur Verfügung steht. Sämtliche Befehle zur Neudefinition von Farben können benutzt werden. Allerdings können Sie den erstellten Entwurf noch nicht sichern.

MODUL 3.1.3

```
13000 REM#******************************
13010 REM BILD SPEICHERN
13020 REM*******************************
13030 GET T$:IF T$="" THEN 13030
13040 IF T$<>"1" AND T$<>"2"THEN RETURN
13050 CO(VAL(T$)*2-2)=PEEK(211):CO(VAL(T
$)*2-1)=INT((CC-1024)/40):RETURN
13060 REM*******************************
13070 CT=CC:POKE 646,CU
13080 IF CO(0)<=CO(2) AND CO(1)<=CO(3) T
HEN 13110
13090 PRINT "SRECHTECK FALSCH DEFINIERT.
";:FOR I=1 TO 1000:NEXT
13100 PRINT "S
```

```
               ";:RETURN
13110 IF (CO(2)-CO(0)-1)*(CO(3)-CO(1)-1)
<251 THEN 13140.
13120 PRINT "BILD ZU GROSS.";:FOR I=1 T
O 1000:NEXT
13130 PRINT "
               ";
13140 FOR I=1 TO 2:PT=1024+40*CO(I*2-1)+
CO(I*2-2):TC(I)=PT:TC(I+2)=PEEK(PT)
13150 POKE PT,42:POKE PT+54272,CU:NEXT
13160 INPUT "SIND DIESE PUNKTE RICHTIG
(J/N):";Q$:IF Q$="J" THEN 13170
13162 PRINT "
     ":FOR I=1 TO 2:POKE TC(I),TC(I+2):NE
XT
13164 RETURN
13170 D1$="":D2$="":FOR I=CO(1)+1 TO CO(
3)-1:FOR J=CO(0)+1 TO CO(2)-1
13180 D1$=D1$+CHR$(PEEK(1024+40*I+J)):PO
KE 1024+40*I+J,42
13190 D2$=D2$+CHR$(PEEK(55296+40*I+J)):P
OKE 55296+40*I+J,0:NEXT J,I:PRINT "";
13200 D3$=D1$:D4$=D2$:FOR I=CO(1)+1 TO C
O(3)-1:FOR J=CO(0)+1 TO CO(2)-1
13210 POKE 1024+40*I+J,ASC(LEFT$(D3$,1))
:D3$=RIGHT$(D3$,LEN(D3$)-1)
13220 POKE 55296+40*I+J,ASC(LEFT$(D4$,1)
):D4$=RIGHT$(D4$,LEN(D4$)-1):NEXT J,I
13230 PRINT "DIES WIRD ABGESPEICHERT.":
FOR I=1 TO 1000:NEXT
13240 INPUT "KASSETTE EINLEGEN, DANN [R
EVERS EIN]RETURN[REVERS AUS]:";Q$
13250 PRINT "
               ";
13260 OPEN 1,1,1,"ARTIST":FOR I=0 TO 3:P
RINT#1,CO(I):NEXT:PRINT#1,LEN(D1$)
13270 FOR I=1 TO LEN(D1$):PRINT#1,ASC(MI
D$(D1$,I,1)) R$ ASC(MID$(D2$,I,1)):NEXT
13280 CLOSE1:RETURN
```

Mit diesem Modul soll ein kleiner Entwurf auf dem Bildschirm definiert und dann gesichert werden.

Kommentar

Zeilen 13030-13050: Wurde im vorhergehenden Modul MODUS 6 gesetzt und danach R betätigt, so nehmen diese drei Zeilen die Eingabe eines weiteren Zeichens auf, bei dem es sich um eine 1 oder 2 handeln muß. Wird 1 betätigt, so wird das aktuelle Cursor-Quadrat als obere linke Ecke des zu sichernden Rechtecks definiert. 2 definiert die untere rechte Ecke. Diese beiden Quadrate liegen in Wirklichkeit außerhalb des zu sichernden Entwurfes und definieren einen äußeren Rahmen um den zu sichernden Bereich. Die Speicherpositionen der beiden Entwurfsecken werden in Feld CO gespeichert. Dieses Feld mußte nicht dimensioniert werden, da es über weniger als zehn Elemente verfügt - durch einfache Eingabe eines Wertes in dieses Feld wird es ausreichend erstellt. CO(0) oder CO(2) wird auf die Position des Cursors in der Zeile gesetzt, wie durch PEEK(211) angegeben. CO(1) oder CO(3) wird auf die aktuelle Nummer der Bildschirmzeile + (aktuelle Speicherposition-Bildschirmbeginn)/40 gesetzt.

Zeilen 13060-13280: Mit diesen Zeilen kann das vorher definierte Rechteck gesichert werden.

Zeile 13070: Die Druckfarbe wird vorübergehend auf die Farbe des Cursors festgelegt.

Zeile 13080: Hier wird geprüft, ob ein gültiges Rechteck definiert wurde, d.h. ein Rechteck, bei dem Länge und Breite angegeben wird. Ist dies nicht der Fall, so wird eine Fehlermeldung ausgedruckt.

Zeile 13110: Die Daten für den Entwurf werden vorübergehend in einer Zeichenfolge gespeichert, so daß geprüft wird, ob die Zeichenfolge nicht zu lang ist. Eine Fehlermeldung wird ausgedruckt, wenn die Zeichenfolge voraussichtlich zu lang ist.

Zeilen 13140-13160: Mit den in Feld CO stehenden Koordinaten werden die Ekken des Rechtecks mit POKE erneut auf den Bildschirm gesetzt. Der Benutzer wird gebeten, die Richtigkeit des zu speichernden Rechteckes zu bestätigen.

Zeile 13170: Die beiden benutzten Zeichenfolgen werden initialisiert. Die beiden Schleifen werden so kombiniert, daß J Zeichen (die Breite des Entwurfs) vom Bildschirm für I Zeilen (die Höhe des Entwurfs) gelesen werden.

Zeilen 13180-13190: Der Bildschirm- und der Farbspeicher werden auf der Grundlage der von den Schleifen angegebenen Adressen mit PEEK überprüft. Die Werte werden in Form von Zeichen mit diesem Codewert zu der im Speicher stehenden Zeichenfolge addiert. Die beiden auf diese Weise gebildeten Zeichenfolgen würden beim Ausdrucken keinen Sinn ergeben. Sie bieten einfach die Möglichkeit, eine Reihe von Werten vorübergehend zu speichern, ohne komplizierte Zeiger zu einer Position in einem Feld setzen zu müssen. Nachdem dies geschehen ist, wird ein Sternchen mit POKE in die Bildschirmadresse gesetzt. Die Farbeigenschaften werden auf schwarz gesetzt, so daß die Verarbeitung des Entwurfs verfolgt werden kann.

Zeile 13200: Eine Kopie von D1$ und D2$ wird genommen. Danach werden zwei weitere Schleifen dazu benutzt, die gespeicherten Zeichen zusammen mit ihren Farbeigenschaften wieder mit POKE auf den Bildschirm zu setzen. Jedesmal, wenn ein Zeichen wieder mit POKE auf den Bildschirm gesetzt wird, wird das jeweils linksbündigste Zeichen aus den beiden Zeichenfolgen abgeschnitten, so daß stets das erste Zeichen der Zeichenfolge benutzt wird. Aufgrund dieses Abschneidens muß eine temporäre Kopie der beiden Originalzeichenfolgen erstellt werden. Diese beiden Schleifen sollen dem Benutzer einfach beweisen, daß der Entwurf richtig gespeichert wird.

Zeilen 13240-13280: Der Entwurf wird auf Band gespeichert.

Zeile 13260: Die Werte in Feld CO werden zusammen mit der Länge von D1$ gespeichert (hier handelt es sich gleichzeitig auch um die Länge von D2$).

Zeile 13270: Eine Schleife, die der Länge von D1$ entspricht, speichert die Werte der Zeichen beider Zeichenfolgen (d.h. der aus dem Bildschirm- und Farbspeicher stammenden Werte). Bedauerlicherweise können die beiden Zeichenfolgen selbst nicht auf Band gespeichert werden, da sie unter Umständen nicht ausdruckbare Zeichen enthalten, die der C64 nicht in Form einer Zeichenfolge speichern kann.

Testen von Modul 3.1.3

Nun sollte ein Entwurf auf Band gespeichert werden. Fällt die erneute Anzeige des Entwurfes zufriedenstellend aus, so wurde die Speicherung höchstwahrscheinlich richtig ausgeführt. Dies kann jedoch nur voll getestet werden, wenn später mindestens das entsprechende Modul des Programms 'Words' eingegeben wird, mit dem auf diese Weise erstellte Entwürfe benutzt werden.

MODUL 3.1.4

```
14000 REM#******************************
14010 REM BILD ABSPEICHERN
14020 REM*******************************
14030 PRINT "■KASSETTE EINLEGEN, DANN [R
EVERS EIN]RETURN[REVERS AUS]:";Q$
14040 PRINT "■
           ";:OPEN 1,1,1,"BILD"
14050 FORI=0TO999:PRINT#1,PEEK(1024+I):P
RINT#1,PEEK(55296+I):NEXT:CLOSE1:RETURN
```

Wird während der Ausführung von Modul 2 MODUS 6 festgelegt und danach S betätigt, so gewährleistet dieses Modul, daß der ganze Bildschirminhalt auf Band gespeichert wird. Dies geschieht völlig unkompliziert, indem der Inhalt des Speichers von Adresse 1024 bis 2023 (Bildschirmspeicher) plus den entsprechenden Farbspeicher-Adressen mit PEEK überprüft wird und die Werte gespeichert werden. Ein späteres Programm kann dann die Werte von Band lesen und sie wieder mit POKE in dieselben Adressen setzen.

Zusammenfassung

Dieses Programm vermittelt viel Freude. Das wichtigste ist jedoch, daß es dem Benutzer die Möglichkeit gibt, komplexe Grafiken problemlos zu erstellen, sie nach Belieben zu bearbeiten und einfach zur Benutzung in nachfolgenden Programmen aufzurufen. Aufgrund der hier benutzten Techniken sollte der Benutzer auch keine Schwierigkeiten mit selbstgeschriebenen Programmen haben, bei denen der Bildschirm- und Farbspeicher mit POKE überprüft werden muß.

Ein Schritt weiter

1. Die Speicherung der Hintergrundfarbe des Bildschirms ist nicht vorgesehen - diese Funktion kann jedoch problemlos hinzugefügt werden.
2. Warum soll nicht auf der untersten Bildschirmzeile angezeigt werden, welcher Modus gerade gesetzt ist.

Artist: Tabelle mit Ein-Tasten-Befehlen

f1	Ermöglicht die Neudefinition der Farbe des Zeichens unter dem Cursor
f2	Ermöglicht die Neudefinition der Bildschirmfarbe
f3	Ermöglicht die Änderung der Druckfarbe
f4	Löscht den aktuellen Entwurf

f5	Setzt RVS (inverse Anzeige) bzw. setzt sie zurück
f6	SAVE-Modus
f7	Pseudo-Modus
f8	Ermöglicht die Änderung der Cursorfarbe
→	Ermöglicht die Eingabe der zweiten im Modus 1-3 festgelegten Farbe

SAVE-Modus:

R	Hier definiert 1 oder 2 die Ecke des zu speichernden Rechtecks
D	Speichert einen Entwurf in kleinem Maßstab
S	Speichert den ganzen Bildschirm

3.2 CHARACTERS

Wie gut auch der Zeichenvorrat in einem Homecomputer sein mag, es wird immer ein Zeitpunkt kommen, an dem das gewünschte Zeichen nicht zur Verfügung steht. Möglicherweise soll in einer anderen Sprache gedruckt und sollen Zeichen mit Akzenten benutzt werden, werden seltene mathematische Symbole benutzt oder wird ein Sonderzeichen benötigt, um dem neuesten Computerspiel den letzten Schliff zu geben. Wie auch immer, der C64 wird dieser Anforderung mit den vom Benutzer definierten Zeichen gerecht.
Beim Starten des Commodore 64 werden sämtliche potentiellen Zeichen in dem Nur-Lese-Speicher (ROM) in einem Abschnitt gespeichert, der bei Adresse 54248 beginnt. Jedes Zeichen nimmt die Form von acht Bytes im Speicher an. Das 8*8 Punktraster, aus dem jedes Zeichen besteht, wird durch die einzelnen Bits der acht für jedes Zeichen reservierten Bytes dargestellt. Wären die acht Bytes im Speicher für ein bestimmtes Zeichen beispielsweise die Bytes 128, 64, 32, 16, 8, 4, 2 und 1, so würden diese in binärer Schreibweise folgendermaßen aussehen: 10000000, 01000000, 00100000, 00010000, 00001000, 00000100, 00000010 und 00000001. Nun werden diese Werte in einen Raster gesetzt:

10000000
01000000
00100000
00010000
00001000
00000100
00000010
00000001

Die gesetzten (oder eingeschalteten) Bits definieren ein Zeichen (in diesem Fall eine diagonale Linie), wobei jedes gesetzte Bit auf dem Bildschirm in ein Bildelement (ein sogenanntes Pixel) umgesetzt wird.
Dies ist alles gut und schön, da die Zeichendaten jedoch im Nur-Lese-Speicher gespeichert sind, können sie nicht geändert werden - sie werden permanent gesetzt, wenn der ROM-Chip gefertigt wird. Glücklicherweise bietet der C64 eine Möglichkeit, dieses Problem zu umgehen. Um dies zu verstehen, muß jedoch als erstes die Methode untersucht werden, mit der die Bildschirmanzeige generiert wird.
Sämtliche Aufgaben, die sich auf den Bildschirm bei dem Commodore 64 beziehen, werden über einen separaten Chip verarbeitet, den 6567 Video-Interface-Chip (oder VIC II Chip). Dieses Arbeitspferd übernimmt sowohl den Bildschirm selbst als auch die Zeichen, die auf den Bildschirm gesetzt werden müssen, wobei ein Speicherbereich definiert wird, in den die Bildschirminformationen gespeichert werden, und ein weiterer Speicherbereich, der für die Zeichendaten benutzt wird.
Entgegen eventueller Annahmen nimmt der VIC II die Zeichendaten nicht aus dem ROM bei Adresse 53248. Der VIC II kann nämlich nur 16K des Speichers gleichzeitig aufnehmen. Befindet sich der Bildschirmspeicher nun in seiner normalen Position bei Adresse 1024-2023 im Speicher, so müssen die Zeichendaten aus einer anderen Stelle zwischen 0 und 16383 im Speicher genommen werden. Um dies zu erreichen, benutzt das Betriebssystem einen kleinen Trick und macht den VIC II glauben, daß in
den Adressen 4096-6143 eine Kopie der Zeichendaten steht. Wann immer der VIC II diesen Speicherbereich betrachtet, entdeckt er die Daten des Zeichenvorrats, obwohl in Wirklichkeit in diesem Speicherbereich wahrscheinlich ein Basic-Programm steht.
Dies mag etwas seltsam erscheinen, ist jedoch von größter Bedeutung, da der VIC II nach den Zeichendaten nicht in dem ROM sucht, der nicht geändert werden kann, sondern in dem Lese-/Schreibspeicher (RAM), d.h. in dem Speicher, zu dem der Benutzer Zugriff hat und den er ändern kann. Ganz so einfach ist dies natürlich nicht. Wie schon gesagt, sieht der VIC II, wenn er den Speicherbereich ab Adresse 4096 betrachtet, nicht wirklich diese Adressen, sondern ein Abbild der Zeichendaten im ROM. Glücklicherweise umfaßt diese Einrichtung nur zwei Speicherblöcke innerhalb des 16K-Blockes, d.h. 4096-6143 und 6144-8193. Wird der VIC II angewiesen, seine Zeichendaten aus einem der anderen 2K Blöcke innerhalb des gesamten 16K-Blockes zu suchen, so sieht er nicht das ROM-Abbild, sondern nimmt die Daten, die tatsächlich im Speicher stehen, und behandelt sie so, als wären sie der Zeichenvorrat.
Nun stellt sich die Frage, welcher Block angegeben werden soll. Der erste zur Verfügung stehende Block ist der Block 2048-4095. Er hat jedoch den Nachteil,

daß an dieser Stelle das Basic-Programm beginnt. Werden in diesen Block neue Zeichendaten mit POKE gesetzt, so kommt es zu einem Programmabsturz. Wir könnten die Blöcke bei 8192, 10240, 12288 oder 14336 benutzen. Dies würde jedoch bedauerlicherweise bedeuten, daß der für das Basic-Programm zur Verfügung stehende Bereich recht drastisch eingeschränkt werden müßte, da ansonsten die Gefahr besteht, daß ein großes Programm den für die Zeichen benutzten Bereich überschreibt. Das Problem wird hier dadurch gelöst, daß der ganze Bereich, den der VIC II Chip adressiert, weiter nach oben an den Anfang des Speichers geschoben wird.
Wie schon gesagt, kann der VIC II einen 16K umfassenden Speicherblock gleichzeitig betrachten, allerdings ist er nicht wählerisch, um welchen 16K-Block es sich dabei handelt. Es gibt vier derartige Blöcke, die bei 0, 16384, 32768 und 49152 beginnen. Wird zu dem Block bei 49152 gegangen und wird der maximale Speicherplatz für Basic berücksichtigt, so steht an dieser Stelle der ROM. Deshalb adressiert der VIC II den 16K Block ab Adresse 32768. Nachdem dies alles geschehen ist, braucht nur noch angegeben zu werden, aus welcher Stelle in diesem Block die Zeichendaten genommen werden und wo die Bildschirmdaten stehen. Auf diese Weise steht einem Basic-Programm ein Speicherplatz von 30K zur Verfügung (2048-32767), während der Bereich für den vom Benutzer definierten Zeichenvorrat in dem RAM über 32768 steht.
Keine Frage, all dies scheint äußerst kompiiziert. Aufgrund der flexiblen Speicherstruktur des C64 ist dies jedoch in Wirklichkeit nur eine Frage einiger weniger POKE-Befehle und schon ist das Problem gelöst. Aufgrund der von den POKE-Befehlen vorgenommenen Änderungen kann der Benutzer mit dem folgenden Programm den gesamten oder einen Teil des Zeichenvorrats des Commodore 64 vollständig neu definieren und den neuen Zeichenvorrat so speichern, daß er von anderen Programmen aufgerufen und benutzt werden kann.

Characters: Variablentabelle

A$	Ein-Tasten-Befehl, der durch Benutzung von GET ermöglicht wird
C1	Ursprüngliche Bildschirmfarbe bei Adresse CC
CC	Aktuelle Position des blinkenden Cursors im Bildschirmspeicher
CH	Nummer des aktuellen Zeichens im Zeichenvorrat
CP	Zeiger zu der Speicheradresse des Zeichens CP
MM	Zur Änderung von CP eingegebener Wert
P1	Zeile, in der der Cursor auf dem Bildschirm steht
P2	Spalte, in der der Cursor auf dem Bildschirm steht
PP	Ursprünglicher Bildschirminhalt bei Adresse CC
TT% (7,7)	Ermöglicht die Verarbeitung von Daten für das aktuelle Zeichen

MODUL 3.2.1

```
11000 REM#*****************************
11010 REM INITIALISIERUNG
11020 REM******************************
11030 POKE 53281,6:PRINT CHR$(142)
11040 POKE 52,128:POKE56,128
11050 POKE 56334,PEEK(56334)AND 254
11060 POKE 1,PEEK(1) AND 251
11070 FOR I=0 TO 2047:POKE 32768+I,PEEK(
53248+I):NEXT
11080 POKE 1,PEEK(1)OR4
11090 POKE 56334,PEEK(56334)OR1
11100 POKE56578,PEEK(56578)OR3
11110 POKE 56576,(PEEK(56576)AND252)OR1
11120 POKE 648,136
11130 POKE 53272,32
```

Mit diesem Modul sollen sämtliche oben aufgeführten Änderungen an der Speicherstruktur vorgenommen und ein ursprünglicher Zeichenvorrat in den angegebenen RAM-Bereich kopiert werden.

Kommentar

Zeile 11030: Mit dieser Zeile wird das System in den Großschreibe-Modus versetzt, da nur der erste der beiden bei dem C64 zur Verfügung stehenden Zeichenvorräte benutzt werden kann, sobald der VIC II das ROM-Abbild nicht mehr betrachtet.

Zeile 11040: Diese beiden POKE-Befehle legen den Anfang des für das Basic-Programm zur Verfügung stehenden Bereichs so fest, daß kein eingegebenes Programm mit dem für die Zeichen reservierten Speicherbereich kollidieren kann. Bei dieser Einstellung steht ein Speicherplatz von 30K zur Verfügung.

Zeilen 11050-11060: Diese beiden POKE-Befehle schalten die Tastaturprüfung ab, so daß keine Unterbrechungen den nächsten Abschnitt des Programms stören können. Danach machen sie den ROM-Zeichenvorrat für das Programm sichtbar, indem das normale Ein-und Ausgabeverfahren ausgeschaltet wird. Während der folgenden Schleife kann das Programm nur gestoppt werden, indem das System ausgeschaltet wird.

Zeile 11070: Mit dieser Zeile wird der Zeichenvorrat von dem ROM in den Speicherbereich ab 32768 kopiert - dies umfaßt die Übertragung von 2K Bytes.

Zeilen 11080-11090: Mit diesen Zeilen wird das normale Ein-/Ausgabeverfahren wieder eingeschaltet, und werden die normalen Unterbrechungen wiederhergestellt.

Zeilen 11100-11110: Diese beiden POKE-Befehle bereiten als erstes den VIC II Chip für eine Änderung des Speicherblockes vor und geben dann Block 1 (32768-49151) an.

Zeile 11120: Diese Adresse liegt außerhalb des VIC II Chips und gibt dem Betriebssystem an, an welcher Stelle der Bildschirmspeicher gespeichert werden soll - in diesem Fall ab 256*13634816.

Zeile 11130: Die Adresse, bei der der VIC II sowohl nach den Bildschirm- als auch nach den Zeichendaten innerhalb des 16K-Blockes sucht, wird vom Inhalt der Adresse 54272 bestimmt - die vier oberen Bits für die Bildschirmdaten und die vier unteren Bits für den Zeichenvorrat. Mit diesem POKE werden die vier oberen Bits auf 0010 gesetzt, d.h. der 1K-Block für die Bildschirmdaten beginnt ab 32768 + 2048, und die vier unteren Bits auf 0000. Damit wird angegeben, daß Zeichendaten aus der Adresse 32768 + 0 genommen werden. Um zu anderen möglichen Adressen in dem 16K-Block zu gelangen, müßte folgende Formel für POKE angegeben werden: ((BILDSCHIRMANFANG—BLOCKANFANG)/1024*16+ (ZEICHEN—BLOCKANFANG)/2048). Der Bildschirm kann nur bei einer 1K-Grenze innerhalb des Blockes beginnen, und die Zeichen nur bei einer 2K-Grenze. Hier wird darauf hingewiesen, daß der Bildschirmspeicher bei 1024 und der Zeichenspeicher bei 4096 hätte belassen werden können. Allerdings sieht der VIC II in diesem 16K umfassenden Speicherblock wie in dem Block, der bei Adresse 0 beginnt, ein ROM-Abbild ab 4096.

Testen von Modul 3.2.1

Der Test dieses Moduls ist recht einfach. Das Modul wird ausgeführt und das System ist während einer bestimmten Zeit gesperrt - es kann auf keine Weise unterbrochen werden. Wird READY auf dem Bildschirm angezeigt, so sollte sich nichts geändert haben - dies zeigt, daß das Modul einwandfrei ausgeführt wurde Ist dies nicht der Fall, so werden auf dem Bildschirm sinnlose Zeichen angezeigt.

MODUL 3.2.2

```
12000 REM#******************************
12010 REM ZEICHEN DARSTELLEN
12020 REM*******************************
12030 CH=0:DIM TT%(7,7)
12040 PRINT "█[HELLBLAU]";:FOR I=1 TO 8:
PRINT "[REVERS EIN]█████████ ":NEXT
12050 PRINT "[REVERS EIN]           "
12060 CP=32768+CH*8
12070 PRINT "█[GRUEN]";:FOR I=CP TO CP+7
:FOR J=7 TO 0 STEP -1
12080 IF (PEEK(I) AND 2↑J)=2↑J THEN PRIN
T "[REVERS EIN] ";
12090 IF (PEEK(I) AND 2↑J)=0 THEN PRINT
"[REVERS AUS] ";
12100 NEXT J:PRINT:NEXT I
12110 PRINT "[HELLBLAU]██NUMMER DES ZEIC
HENS:";CH
12120 INPUT "█NEUE NUMMER (0 FUER UMDEFI
NIERUNG):";MM:CH=CH+MM
12130 IF CH<0 THEN CH=0
12140 IF CH>255 THEN CH=255
12150 IF MM=0 THEN 13000
12160 GOTO 12040
```

Neue Zeichen können auf verschiedene Art und Weise in den Zeichenspeicher eingegeben werden. Sie können gegebenenfalls in einem 8*8-Raster gezeichnet werden. Danach können die Linien mit den Punkten in Binärform und schließlich in Dezimalform umgewandelt werden, die Zahlen als Datenanweisungen eingegeben und dann mit POKE in den Speicher gesetzt werden. Glücklicherweise führt der C64 diese Aufgabe wesentlich einfacher aus, indem er das aktuelle Zeichenraster auf dem Bildschirm zeichnet, das dann problemlos verarbeitet werden kann. Mit diesem Modul wird das Zeichenraster gezeichnet, während es mit dem nächsten verarbeitet werden kann.

Kommentar

Zeilen 12040-12050: In der oberen linken Ecke des Bildschirms wird ein 8*8-Kästchen gezeichnet.

Zeile 12060: Die Position der aktuellen Zeichendaten wird berechnet.

Zeilen 12070-12100: Eine vergrößerte Version der aktuellen Zeichen wird in dem neu gezeichneten Kästchen ausgedruckt. Hier wird auf die Benutzung des AND verwiesen, um den Inhalt der einzelnen Bits innerhalb der acht Bytes im Speicher für das Zeichen zu erhalten. Hier besteht die einzige Funktion von AND darin, zwei Binärzahlen miteinander zu vergleichen und eine dritte Zahl zu erzeugen, bei der nur die Bits gesetzt sind, die auch in den beiden ursprünglich miteinander verglichenen Zahlen gesetzt waren. Wird 193 somit durch AND mit 129 verknüpft (Binärform 11000001 AND 1000001), so ist das Ergebnis 129, da Bit 6 in der ersten Zahl nicht auch in der zweiten gesetzt ist. Wird der Wert eines Bytes durch AND mit 2↑ (J) verknüpft, wobei J von 0 bis 7 geht, so wird angegeben, ob Bit J gesetzt ist oder nicht - hier wird nochmals darauf hingewiesen, daß die Bits von 0 bis 7 von rechts nach links numeriert sind.

Zeilen 12100-12160: Die Nummer des angezeigten Zeichens wird angegeben. Der Benutzer hat die Möglichkeit, den Zeichenzeiger innerhalb der 255 Zeichen hin- und herzubewegen. Wird Null eingegeben, so geht das Programm zum nächsten Modul weiter.

Testen von Modul 3.2.2

Auch dieser Test ist recht einfach. Wurde das Modul richtig eingegeben, so wird durch Ausführung des Programms (nach einer Pause) eine vergrößerte Version von'@'auf dem Bildschirm ausgedruckt. Außerdem sollten auch die anderen Zeichen angezeigt werden können.

MODUL 3.2.3

```
13000 REM#*****************************
13010 REM ZEICHEN UMDEFINIEREN
13020 REM******************************
13030 PRINT "□
                ":REM 40 SPACES
13040 PRINT "[WEISS]'I' INVERTIEREN",,,"
[GRUEN]'M' ZUM SPIEGELN",,,"[WEISS]'R' Z
URUECK"
13050 PRINT "[GRUEN]'1' PUNKT SETZEN":PR
INT "[WEISS]'0' PUNKT LOESCHEN"
13060 PRINT "[GRUEN]'T' DREHEN",,,"[WEIS
S]'P' IN DEN SPEICHER SCHREIBEN
```

```
13070 PRINT "[GRUEN]'D' ZEICHENSATZ ABSP
EICHERN":PRINT"'C' ZEICHENSATZ LADEN"
13080 PRINT "[GRUEN]'N' SPEICHER NORMALI
SIEREN UND BEENDEN
13100 PRINT "[WEISS] CURSOR-TASTEN UM DE
N CURSOR ZU BEWEGEN"
13110 PRINT "▣";
13120 GET A$
13130 CC=PEEK(211)+PEEK(210)*256+PEEK(20
9):PP=PEEK(CC):C1=55296+CC-34816
13140 C2=PEEK(C1)
13150 POKE CC,42:POKE C1,1
13160 FOR I=1 TO 15:NEXT:POKE CC,PP:POKE
 C1,C2:IF A$="" THEN 13120
13170 P1=INT((CC-34816)/40):P2=CC-(34816
+40*P1)
13180 IF (P1>0 AND A$="□") OR (P1<7 AND
A$="▣") THEN PRINT A$;:GOTO 13120
13190 IF (P2>0 AND A$="▌") OR (P2<7 AND
A$="▐") THEN PRINT A$;:GOTO 13120
13200 IF A$="1" THEN PRINT "[GRUEN][REVE
RS EIN] ";
13210 IF A$="0" THEN PRINT "[BLAU][REVER
S AUS] [HELLBLAU]";
13220 IF A$<>"I" THEN 13280
13230 FOR I=0 TO 7:FOR J=0 TO 7
13240 IF PEEK(34816+I*40+J)=32 THEN 1326
0
13250 POKE 34816+40*I+J,32:POKE 55296+40
*I+J,182:GOTO 13270
13260 POKE 34816+I*40+J,160:POKE 55296+4
0*I+J,181
13270 NEXT J,I
13280 IF A$="R" THEN GOTO 12040
13290 IF A$<>"M" THEN 13370
13300 FOR I=0 TO 7:FOR J=0 TO7:TT%(I,J)=
0:NEXT J,I
13310 FOR I=0 TO 7:FOR J=0 TO 7
13320 IF PEEK(34816+40*I+J)=160 THEN TT%
(I,J)=1
```

```
13330 NEXT J,I
13340 PRINT"■";:FORI=0TO7:FOR J=7 TO 0ST
EP-1:IF TT%(I,J)=1 THEN PRINT "[GRUEN][R
EVERS EIN] [REVERS AUS]";
13350 IF TT%(I,J)=0 THEN PRINT "[BLAU] "
;
13360 NEXT J:PRINT:NEXT I
13370 IF A$<>"T" THEN 13450
13380 FOR I=0 TO 7:FOR J=0 TO 7:TT%(I,J)
=0:NEXT J,I
13390 FOR I=0 TO 7:FOR J=0 TO 7
13400 IF PEEK(34816+40*I+J)=160 THEN TT%
(7-J,7-I)=1
13410 NEXT J,I
13420 PRINT"■";:FORI=0TO7:FOR J=7 TO 0ST
EP-1:IF TT%(I,J)=1 THEN PRINT "[GRUEN][R
EVERS EIN] [REVERS AUS]";
13430 IF TT%(I,J)=0 THEN PRINT "[BLAU] "
;
13440 NEXT J:PRINT:NEXT I
13450 IF A$<>"P" THEN 13510
13460 :FOR I=0 TO 7:TT%(0,I)=0:NEXT
13470 FOR I=0 TO 7:FOR J=0 TO 7
13480 IF PEEK(34816+40*I+J)=160 THEN TT%
(0,I)=TT%(0,I) OR 2↑(7-J)
13490 NEXT J,I
13500 FOR I=0 TO 7:POKE CP+I,TT%(0,I):NE
XT:GOTO 12040
13510 IF A$<>"D" THEN 13550
13520 OPEN 1,1,1,"ZEICHENSATZ"
13530 FOR I=0 TO 2047:T%=PEEK(32768+I):P
RINT#1,T%:NEXT
13540 CLOSE 1
13550 IF A$<>"C" THEN 13590
13560 OPEN 1,1,0,"ZEICHENSATZ"
13570 FOR I=0 TO 2047:INPUT#1,T:POKE 327
68+I,T:NEXT
13580 CLOSE1
13590 IF A$<>"N" THEN 13660
13600 POKE 52,160:POKE56,160:CLR
```

```
13610 POKE56578,PEEK(56578)OR3
13620 POKE 56576,(PEEK(56576)AND252)OR3
13630 POKE 53272,21
13640 POKE 648,4
13650 END
13660 GOTO 13120
```

Dieses Modul führt eine Reihe von Funktionen im Zusammenhang mit der Verarbeitung der Zeichen auf dem Bildschirm aus, wobei diese unter anderem neu definiert, wieder in den Speicher gesetzt und mit SAVE auf Band gesichert werden können.

Kommentar

Zeilen 13030-13100: Kurze Instruktionen zur Benutzung des Moduls werden auf dem Bildschirm ausgedruckt.

Zeilen 13120-13160: Dieses Modul birgt keine Überraschungen. Hier handelt es sich einfach um die Routine für den blinkenden Cursor aus dem Artist Programm.

Zeile 13170: Cursorposition: P1 entspricht der Zeilennummer und P2 der Spaltennummer, wobei die Zeilen von oben nach unten und die Spalten von links nach rechts auf dem Bildschirm verlaufen.

Zeilen 13180-13190: Begrenzt die Cursorbewegung mit dem 8*8-Quadrat.

Zeilen 13200-13210: Wird 1 betätigt, so wird ein grünes Quadrat eingezeichnet. Wird 0 betätigt, so wird ein Quadrat gelöscht.

Zeilen 13230-13270: Diese beiden Schleifen durchsuchen das Quadrat, wobei die eingezeichneten oder leeren Elemente umgekehrt werden, so daß ein inverses Zeichen erzeugt wird.

Zeile 13280: Durch Eingabe von R wird zum vorhergehenden Modul zurückgegangen.

Zeilen 13290-13360: Diese Routine erzeugt ein Spiegelbild des Rasters - das Zeichen wird scheinbar von hinten betrachtet.

Zeile 13300: Das ganzzahlige Feld TT % wird gelöscht.

Zeilen 13310-13330: Der Bildschirminhalt wird in das Feld übertragen. Der Bild-

schirm kann nicht direkt verarbeitet werden, da dadurch ein Quadrat von links nach rechts übertragen und dann zwei Mal gelesen werden könnte, so daß ein sinnloses Ergebnis entsteht.

Zeilen 13340-13360: Nachdem der Inhalt des Rasters in das Feld übertragen wurde, werden die Informationen nun wieder auf den Bildschirm gelesen. Das horizontale Element wird jedoch umgekehrt, so daß Position 7 in Position 0 gesetzt wird.

Zeilen 13370-13440: Der Inhalt des Rasters wird um 90 entgegen dem Uhrzeigersinn gedreht.

Zeilen 13420-13440: Der Inhalt das Feld wird wieder entgegen dem Uhrzeigersinn auf den Bildschirm gesetzt - so wird Position 0,7 zu Position 0,0 und Position 0,0 zu Position 7,0.

Zeilen 13450-13500: Das neu definierte Zeichen wird wieder in den Zeichenspeicher gesetzt. Es wird nun zu einem festen Bestandteil des vom Benutzer definierten Zeichenvorrats.

Zeile 13460: Da nur acht Bytes für jedes Zeichen erforderlich sind, müssen nur acht Bytes des Feldes, Zeile Null, 0-7, gelöscht werden.

Zeilen 13470-13490: Jede Zeile der Feldmatrix wird durchsucht. Wird ein eingezeichnetes Quadrat entdeckt, so wird es in ein einzelnes Bit in einem der acht Bytes übersetzt, mit denen das Zeichen definiert wird. Nachdem AND für das Lesen einzelner Bits benutzt wurde, wird hier auf die Benutzung von OR zur Verarbeitung einzelner Bits verwiesen. Werden zwei Binärzahlen mit OR verknüpft, so werden sämtliche Bits, die in einer (oder beiden) Zahlen gesetzt sind, auch in der sich ergebenden Zahl gesetzt. Wird eine Zahl durch OR mit 2↑ (J) verknüpft, wobei J von 0 bis 7 geht, so bedeutet dies, daß Bit J eingeschaltet wird, unabhängig davon, ob es vorher gesetzt war oder nicht.

Zeile 13500: Die acht Bytes des Feldes werden bei der Position in den Speicher gesetzt, die vorher von dem Zeichen belegt wurde, das neu definiert wurde.

Zeilen 13510-13540: Der Speicherbereich, der bei 32768 beginnt, wird in Form von Ganzzahlen auf Band gespeichert.

Zeilen 13550-13560: Ein vorher gespeicherter Zeichenvorrat kann zur weiteren Verarbeitung vom Band geholt werden. Hinweis: Dies ist auch ein Beispiel dafür, wie der neue Zeichenvorrat von einem anderen Programm zur späteren Benutzung aufgerufen werden kann.

Zeilen 13590-13650: Wird das Programm beendet, so muß der Speicher wieder in die Ausgangsbedingung zurückversetzt werden - es sei denn, der neue Zeichenvorrat soll mit einem anderen Programm benutzt werden, das nun geladen wird. Wird der Speicher nicht zurückgesetzt, so bedeutet dies, daß nachfolgenden Programmen 8K Speicherplatz weniger zur Verfügung steht und sie gezwungen sind, den neu definierten Zeichenvorrat zu benutzen.

Zeile 13600: Basic wird wieder auf seine ganze potentielle Größe zurückgesetzt.

Zeilen 13610-13620: Der von VIC II adressierte Speicherblock wird auf 3 zurückgesetzt (0-16383).

Zeilen 13630-13640: Der Anfang des Bildschirmspeichers wird wieder auf 1024 (die normale Position) und der Anfang des Zeichenspeichers auf 4096 zurückgesetzt.

Testen von Modul 3.2.3

Da es sich hier um ein langes Modul mit einer Vielzahl von Funktionen handelt, wird empfohlen, daß jede Funktion bei der Eingabe getestet wird. Ist eine bestimmte Funktion fehlerhaft und wurden Änderungen in eine Zeile eingegeben, so braucht das Programm nicht von Anfang an neu ausgeführt zu werden. In diesem Fall wird einfach GOTO 12000 ausgeführt, da der Zeichenvorrat über dem Basic-Bereich und die Speicherstruktur durch die Eingabe neuer Zeilen nicht gestört werden. Läuft alles einwandfrei, so können die in den Kommentaren beschriebenen Funktionen benutzt werden.

Zusammenfassung

Wie Sie feststellen werden, ist dies schon in sich ein äußerst erfreuliches Programm. Wie leistungsfähig dieses Programm jedoch ist, zeigt sich erst, wenn es zur Belebung der Ausgabe anderer Programme benutzt wird. Da es Basic nicht wirklich verschiebt, sondern nur den zur Verfügung stehenden Platz beschränkt, können neue Programme in das System geladen werden, die den neu definierten Zeichenvorrat ebenfalls benutzen können. Wurde das System seit der Neudefinition des Zeichenvorrats ausgeschaltet oder der Speicher standardisiert, so braucht nur das erste Modul vor alle nachfolgenden Programme hinzugefügt (minus Zeilen 11050-11090) und der neu definierte Zeichenvorrat mit der Routine bei 13560-13580 von Band geladen zu werden. Wird der Buchstabe A als Leerzeichen neu definiert, so muß jedoch daran gedacht werden, daß jedes vom Programm ausgegebene A, selbst in der Programmauflistung, neu definiert wird. Im Hinblick auf die Leserlichkeit ist es im allgemeinen besser, nur die Grafikzeichen neu zu definieren.

Neben den allgemeinen Vorteilen des Programms wurden hier jedoch auch einige der Möglichkeiten vorgestellt, die sich aus der flexiblen Speicherstruktur des C64 ergeben. Außerdem wurden die Techniken angesprochen, mit denen die zur Verfügung stehenden Funktionen maximal ausgenutzt werden können. Möchten Sie weiter in die Speicherverarbeitung vordringen, so sollten Sie das Programmhandbuch 'Alles über den Commodore 64', Band 1 der Sachbuchreihe, erwerben - mit diesem Programm als Rückhalt sollten Sie dann keine Schwierigkeiten haben, die Angaben in diesem Handbuch zu verstehen und anzuwenden.

Ein Schritt weiter

1. Eine einfache Erweiterung des Programms besteht in einer Routine, mit der die Position von zwei Zeichen ausgetauscht, oder mit der ein neu definiertes Zeichen an eine andere Stelle in dem Zeichenvorrat gesetzt werden kann.
2. Es wäre äußerst zeitaufwendig, einen ganzen neuen Zeichenvorrat mit diesem Programm zu erstellen. Warum also nicht einige Befehle zur Blockbearbeitung hinzufügen, mit denen ein ganzer Zeichenvorrat innerhalb bestimmter Grenzen umgekehrt, umgedreht, spiegelbildlich dargestellt werden kann usw. Programmauflistungen sehen äußerst interessant aus, wenn sämtliche Buchstaben auf dem Kopf stehen

3.3 SPRITES

Mit dem Characters-Programm haben wir den Weg für die Überprüfung einer der Funktionen des Commodore 64 geebnet, von denen andere Benutzer von Homecomputern nur träumen können- Sprites. Mit der Einführung des C64 sind die Tage vorbei, in denen nur Programmierer im Maschinencode hochauflösende Entwürfe glatt und problemlos mit einem unglaublichen Realismus über den Bildschirm bewegen konnten. Insbesondere im Bereich der Computerspiele stellen die Sprites eine Revolution bei den Homecomputern der niedrigeren Preisklasse dar.
Im wesentlichen unterscheidet sich ein Sprite nur wenig von den vom Benutzer definierten Zeichen, mit denen wir gerade experimentiert haben. In das Erstellen der Sprite-Funktion wurde viel technisches Vorstellungsvermögen und technische Kompetenz gesteckt, aus der Sicht des Benutzers ist ein Sprite jedoch nur einfach ein größeres Zeichen, das flexibler über den Bildschirm bewegt werden kann.
Wie die Zeichen des normalen Zeichenvorrats werden die Sprites mit einer Reihe von Bytes definiert, die im RAM gespeichert sind. Anstelle eine 8*8-Rasters benutzen Sprites jedoch ein Raster aus 24 mal 21 Punkten. Dies kann natürlich nicht

mit den 64 Bits in acht Bytes definiert werden. In Wirklichkeit wird jede Zeile eines Sprite mit 3 Bytes (24 Bits) definiert. Da 21 Zeilen vorhanden sind, sind insgesamt 63 Bytes zur Definition eines Sprite erforderlich. Sprite-Daten können an einem beliebigen sicheren Platz innerhalb des 16K-Speicherblocks gespeichert werden, der von dem VIC II adressiert wird. Innerhalb dieses Blocks können bis zu acht Sprites gleichzeitig definiert werden. Wesentlich mehr Sprite-Entwürfe können jedoch in Reserve gehalten und gegebenenfalls sofort aktiviert werden.
Die Hauptspeicheradressen, mit denen die Benutzung von Sprites kontrolliert wird, sind:

a) 2040-2047: Diese acht Adressen sind die sogenannten Sprite-Pointer. Sie geben an, aus welcher Stelle in dem 16K-Block die Daten für einen bestimmten Sprite genommen werden müssen. Da Sprites in Blöcken zu 64 Bytes gespeichert sind (auch wenn nur 63 Bytes benutzt werden) kann mit den 256 Werten, die mit POKE in jeden Pointer gesetzt werden können, der gesamte 16K-Block abgedeckt werden. So werden die Daten für Sprite 2 bei 64*PEEK(2042) aus dem Speicher genommen.

b) 53269: Das Register zur Aktivierung der Sprites. Ein Sprite ist nur sichtbar, wenn das entsprechende Bit in diesem Register gesetzt ist.

c) 53248-53264: Die Register für die Sprite-Position. Sie werden paarweise von 53248 bis 53263 benutzt und definieren die X- und Y-Koordinaten der oberen linken Ecke des Sprite-Rasters auf dem Bildschirm. Da jedoch der Bildschirm breiter ist als der Höchstwert, der in einem einzelnen Byte gespeichert werden kann (255 im Vergleich zu 320 Pixels), wird mit einem Bit in Adresse 53264 angegeben, ob die Position jedes Sprite auf der X-Achse größer ist als 255. Dies ergibt insgesamt 512 mögliche Positionen auf der X-Achse und 256 Positionen auf der Y-Achse.

d) 53287-53294: Die Register für die Sprite-Farbe. Jeder Sprite kann eine der 16 Farben des C64 annehmen, indem einfach der richtige Wert mit POKE in das entsprechende Register gesetzt wird. In Wirklichkeit sind mehr Adressen als diese relevant, sie reichen jedoch für den Augenblick aus.

Schließlich muß noch entschieden werden, an welche Stelle die Sprite-Daten gesetzt werden sollen. Werden nur drei Sprites gewünscht, so wird der Kassetten-Eingabe-/Ausgabepuffer empfohlen, der in den Adressen 828 bis 1019 steht. (Natürlich können keine Daten geladen oder gesichert werden, während die Sprites an dieser Stelle stehen.) Werden mehr Sprites gewünscht, so muß ein Speicherbereich für sie reserviert werden, genau wie bei den vom Benutzer definierten Zeichen. Der Abwechslung wegen werden wir für unser Programm zur Definition von Sprites eine andere Lösung wählen, als bei dem Characters-Programm. Wir werden den Anfang des Basic-Programms von 2048 nach 4096 verschieben, so daß uns 2K im Speicher verbleiben, in die wir bis zu 32 separate Gruppen mit Sprite-Daten setzen können. Dies ist in sofern praktisch, als die

Struktur des Bildschirmspeichers in keiner Weise verschoben zu werden braucht - allerdings muß die Basic-Anfangsadresse zurückgesetzt werden, bevor das Programm geladen wird.
Nachdem dies geschehen ist, ermöglicht das Programm genau wie der Zeichen-Generator die Definition und Verarbeitung der Sprite-Raster und bietet die Möglichkeit, die Raster zur Benutzung durch weitere Programme auf Band zu sichern.
Dieses Programm wird am einfachsten eingegeben, indem zuerst Characters geladen und dieses Programm dann angepaßt wird.

Ladeprogramm für Sprites

Die folgenden Zeilen gehören NICHT zu dem Hauptprogramm, sondern sollen in den Commodore 64 eingegeben und auf Band gespeichert werden, bevor das Hauptprogramm eingegeben und gespeichert wird. Dieses Programm soll den Anfang von Basic zurücksetzen und danach das Hauptprogramm in den neu konfigurierten Speicher laden:

```
100 REM*************** *****************
110 REM LOADER
120 REM*************** *****************
130 POKE43,1:POKE 44,16:POKE 4096,0:CLR
140 LOAD "SPRITES"
```

Kommentar

Die Adressen 43 und 44 sind die Zeiger, mit denen das System auf den Anfang des Basic-Programms zeigt, in dem normalerweise die Werte 1 und 8 stehen (Adresse 1 + 256*8 2049). Die Hauptzeile ändert nur diesen Wert in 4097. Das erste Programmbyte muß stets eine Null sein, die mit POKE gespeichert wird. Danach wird der Speicher gelöscht und damit die Neukonfiguration beendet. Nachdem dies geschehen ist, wird das Hauptprogramm automatisch geladen. Selbst auf die Gefahr hin, Sie zu langweilen, möchte ich noch einmal wiederholenn, daß diese Zeilen NICHT Teil des Hauptprogramms sind. Werden sie am Anfang des Hauptprogramms aufgenommen, so werden die ersten 2K des Programms gelöscht, wenn dieses ausgeführt wird.

Sprites: Variablentabelle
(sofern sie von Characters abweichen)

SP	Adresse des aktuellen Sprite-Pointers
SC	Adresse des Registers für die Sprite-Farbe

SS	Anfang der Sprite-Daten
FNS(SN)	Anfang des Datenblockes für Sprite SN
SN	Sprite-Nummer
TT% (20,23)	Feld für die vorübergehende Verarbeitung von Sprite-Daten
MM	Wert, in den SN verschoben wird.

MODUL 3.3.1

```
12000 REM#********************************
12002 REM SPRITE-POINTER SETZEN
12005 REM*********************************
12010 SP=2040:SE=53269:SS=2048
12020 DEF FNS(SN)=SS+64*(SN)
12030 SN=0:DIM TT%(20,23):POKE 53281,6
```

Die hier vereinbarten Variablen werden in der Variablentabelle erläutert.

MODUL 3.3.2

```
13000 REM#*******************************
13010 REM SPRITE DARSTELLEN
13020 REM********************************
13030 POKE 53269,1:POKE 53287,1:POKE SP,
FNS(SN)/64
13040 POKE 53248,0:POKE 53264,1:POKE 532
49,80
13050 PRINT "▟";:FOR I=1 TO 11:PRINT "[H
ELLBLAU][REVERS EIN]
      "
13060 PRINT "[GELB]███████████████████████
   ▒":NEXT
13070 PRINT "□[REVERS EIN][HELLBLAU]
                          "
13080 PRINT "□[HELLROT]▒█▒█▒█▒█▒█▒█▒█▒█▒█
 ▒ ▒ ▒ ▒"
13090 PRINT "▣[GRUEN]";:FOR I=FNS(SN) TO
 FNS(SN)+62 STEP 3:FOR J=0 TO 2
13100 FOR K=7 TO 0 STEP -1
13110 IF (PEEK(I+J) AND 2↑K)=2↑K THEN PR
INT "●";
13120 IF (PEEK(I+J) AND 2↑K)=0 THEN PRIN
```

```
T "  ";
13130 NEXT K,J:PRINT:NEXT I
13140 PRINT "■[HELLGRUEN]SPRITE NUMMER:"
;SN
13150 INPUT "SPRITENUMMER ODER 0 ZUM UMD
EF.:";MM:SN=SN+MM
13160 IF SN<0 THEN SN=0
13170 IF SN>31 THEN SN=31
13180 IF MM=0 THEN POKE 53248,0:POKE 532
64,1:POKE 53249,200:GOTO `14000
13190 GOTO 13000
```

Nahezu identisch mit dem Modul für das Zeichnen des Rasters in Characters.

Kommentar

Zeile 13030: 53269 ist das Register zur Aktivierung der Sprites - mit diesem POKE-Befehl wird das Bit 0 gesetzt und Sprite 0 eingeschaltet. 53287 ist das Farbregister für Sprite 0, und der POKE-Befehl setzt die Farbe auf weiß. Der Pointer von Sprite Null wird so gesetzt, daß er auf den ersten 64 Bytes umfassenden Block in dem reservierten Speicherbereich zeigt.

Zeile 13040: Sprite 0 ist auf der X-Achse auf 256 und auf der Y-Achse auf 80 gesetzt.

Zeilen 13050-13080: Die Umrahmung des Rasters wird ausgedruckt.

Zeilen 13090-13130: Die I-Schleife betrachtet die Sprite-Daten in Dreiergruppen, die J-Schleife betrachtet jedes Byte und die K-Schleife jedes Bit. Für jedes gesetzte Bit wird ein Kreis ausgedruckt.

Zeilen 13140-13190: Der Benutzer kann den Sprite-Pointer verschieben. Hier wird darauf hingewiesen, daß der Speicherbereich, auf den der Sprite-Pointer zeigt und nicht der eigentliche Sprite-Pointer geändert wird. In dem vorliegenden Programm wird stets Sprite 0 benutzt.

Testen von Modul 3.3.2

Bei der Ausführung dieses Moduls (wobei das Ladeprogramm zuerst ausgeführt werden muß) wird ein sinnloser Sprite rechts neben dem Raster angezeigt. Die einzelnen Punkte werden in dem Raster ausgefüllt. Gelegentlich ist die Verbindung zwischen den beiden wegen der automatischen Schattierungen in den Spri-

tes nur schwer zu sehen. Zwei auf derselben Zeile gesetzte Bits, die mit einem Leerzeichen voneinander getrennt werden, werden in Wirklichkeit als ein schwarzer Block angezeigt.

MODUL 3.3.3

```
14000 REM#*****************************
14010 REM SPRITE UMDEFINIEREN
14020 REM******************************
14030 PRINT "□
                ":REM 39 SPACES
14040 F$="███████████████████████████"
14050 PRINT F$;"[WEISS]'I' INVERTIER."
14060 PRINTF$;"[GRUEN]'M' SPIEGELN"
14070 PRINTF$;"[WEISS]'R' ZURUECK"
14080 PRINTF$;"[GRUEN]'1' SETZEN"
14090 PRINTF$;"[WEISS]'0' LOESCHEN"
14100 PRINT F$;"[GRUEN]'T' DREHEN"
14110 PRINTF$;"[WEISS]'P' SPEICHERN"
14120 PRINTF$;"[GRUEN]'D' SAVE'N"
14130 PRINTF$;"[WEISS]'L' LADEN"
14140 PRINTF$;"[GRUEN]'E' ENDE"
14150 PRINT F$;"[WEISS]CURSORTASTEN"
14155 PRINT F$;"[WEISS]ZUM STEUERN"
14160 PRINT "▣";
14170 GET A$
14180 CC=PEEK(211)+PEEK(210)*256+PEEK(20
9):PP=PEEK(CC):C1=55296+CC-1024
14190 C2=PEEK(C1)
14200 POKE CC,42:POKE C1,1
14210 FOR I=1 TO 15:NEXT:POKE CC,PP:POKE
 C1,C2:IF A$="" THEN 14170
14220 P1=INT((CC-1024)/40):P2=CC-(1024+4
0*P1)
14230 IF (P1>0 AND A$="□") OR (P1<20 AND
 A$="▣") THEN PRINT A$;:GOTO 14170
14240 IF (P2>0 AND A$="▌") OR (P2<23 AND
 A$="▐") THEN PRINT A$;:GOTO 14170
14250 IF A$="1" THEN PRINT "[GRUEN]●▌";:
GOTO 14170
14260 IF A$="0" THEN PRINT "[BLAU][REVER
```

```
S AUS] [HELLBLAU]";:GOTO 14170
14270 IF A$<>"I" THEN 14330
14280 FOR I=0 TO 20:FOR J=0 TO 23
14290 IF PEEK(1024+I*40+J)=32 THEN 14310

14300 POKE 1024+40*I+J,32:POKE 55296+40*
I+J,182:GOTO 14320
14310 POKE 1024+I*40+J,81:POKE 55296+40*
I+J,181
14320 NEXT J,I:GOTO 14170
14330 IF A$="R" THEN GOTO 13030
14340 IF A$<>"M" THEN 14420
14350 FOR I=0 TO 20:FOR J=0 TO 23:TT%(I,
J)=0:NEXT J,I
14360 FOR I=0 TO 20:FOR J=0 TO 23
14370 IF PEEK(1024+40*I+J)=160 THEN TT%(
I,J)=1
14380 NEXT J,I
14390 PRINT"■";:FORI=0TO20:FOR J=23TO0 S
TEP-1:IF TT%(I,J)=1 THEN PRINT "[GRUEN][
REVERS EIN] [REVERS AUS][HELLBLAU]";
14400 IF TT%(I,J)=0 THEN PRINT "[BLAU] "
;
14410 NEXT J:PRINT:NEXT I:GOTO 14170
14420 IF A$<>"T" THEN 14500
14430 FOR I=0 TO 20:FOR J=0 TO 23:TT%(I,
J)=0:NEXT J,I
14440 FOR I=0 TO 20:FOR J=0 TO 20
14450 IF PEEK(1024+40*I+J)=81 THEN TT%(2
0-J,20-I)=1
14460 NEXT J,I
14470 PRINT"■";:FORI=0TO20:FOR J=23TO0 S
TEP-1:IF TT%(I,J)=1 THEN PRINT "[GRUEN]●
[HELLBLAU]";
14480 IF TT%(I,J)=0 THEN PRINT "[BLAU] [
HELLBLAU]";
14490 NEXT J:PRINT:NEXT I:GOTO 14170
14500 IF A$<>"P" THEN 14560
14510 FOR I=0 TO 20:FOR J=0 TO 2:TT%(I,J
)=0:NEXT J,I
```

```
14520 FOR I=0 TO 20:FOR J=0 TO 2:FOR K=0
 TO 7
14530 IF PEEK(1024+40*I+J*8+K)=81 THEN T
T%(I,J)=TT%(I,J) OR 2↑(7-K)
14540 NEXT K,J,I
14550 FOR I=0 TO20:FOR J=0 TO 2:POKE FNS
(SN)+I*3+J,TT%(I,J):NEXT J,I:GOTO 13030
14560 IF A$<>"S" THEN 14620
14570 PRINT "SQQQQQQQQQQQQQQQQQQQQQQQQ[G
RUEN]ANZAHL SPRITES:";NN
14580 IF NN<1 OR NN>32 THEN 14570
14590 OPEN 1,1,1,"SPRITES":PRINT#1,NN
14600 FOR I=0 TO NN*64-1:T%=PEEK(SS+I):P
RINT#1,T%:NEXT
14610 CLOSE 1
14620 IF A$<>"L" THEN 14660
14630 OPEN 1,1,0,"SPRITES":INPUT#1,NN
14640 FOR I=0 TO NN*64-1:INPUT#1,T:POKE
SP+I,T:NEXT
14650 CLOSE1
14660 IF A$<>"E" THEN 14680
14670 POKE 53269,0:POKE 43,1:POKE 44,8:P
OKE 2048,0:CLR:END
14680 IF A$<>"X" THEN 14740
14690 INPUT "SQQQQQQQQQQQQQQQQQQQQQQQ[GR
UEN]AUSTAUSCHEN MIT SPRITE NUMMER:";S2
14700 IF S2<0 OR S2>31 THEN 14690
14710 FOR I=FNS(SN) TO FNS(SN)+62:LET T1
=PEEK(I):POKE I,PEEK(I+FNS(SN)-FNS(S2))
14720 POKE I+FNS(SN)-FNS(S2),T1:NEXT
14730 GOTO 13000
14740 IF A$<>"C" THEN 14840
14750 IF PEEK(53276)AND 1=1 THEN POKE 53
276,0:GOTO 14170
14760 INPUT "[GRUEN]SQQQQQQQQQQQQQQQQQQQ
farbe fuer 01 (0-15):";c1
14770 IFC1<0ORC1>15THENPRINT"□
                                ":GOTO 14760
14780 INPUT "[WEISS]SQQQQQQQQQQQQQQQQQQQ
farbe fuer 10 (0-15):";c2
```

```
14790 IFC2<0ORC2>15THENPRINT"□
                                 ":GOTO 14780
14800 INPUT "[GELB]S[illegible]
farbe fuer 11 (0-15):";c3
14810 IFC3<0ORC3>15THENPRINT"□
                                 ":GOTO 14800
14820 POKE 53285,(PEEK(53285)AND240)ORC1
:POKE 53287,(PEEK(53287)AND240)ORC3
14830 POKE 53286,(PEEK(53286) AND 240) O
R C2:POKE 53276,1:GOTO 14170
14840 GOTO 14170
```

Dieses Modul erfüllt denselben Zweck wie das Modul zur Neudefinition der Zeichen im letzten Programm.

Kommentar

Zeilen 14050-14150: Instruktionen zur Benutzung des Moduls.

Zeilen 14170-14210: Standardmodul zur Bewegung des Cursors.

Zeile 14220: Zeile und Spalte des Cursors in dem Raster.

Zeile 14330: Durch R wird zu dem vorhergehenden Modul zurückgegangen.

Zeilen 14340-14490: Drehung im Uhrzeigersinn.

Zeilen 14500-14550: Der neu definierte Sprite wird wieder in den Speicher gesetzt. Die I-Schleife überspringt jede Zeile des Rasters ab. Die J-Schleife prüft in Gruppen von drei Bytes, während die K-Schleife jedes Bit überprüft.

Zeilen 14560-14610: Die Sprite-Daten werden auf Band gesichert. Der Benutzer kann angeben, wie viele Sprites gesichert werden sollen. Dadurch können drei Sprites gesichert werden, die in dem Bandeingabepuffer eines nachfolgenden Programms gespeichert werden können.
Wird ein Sprite vom Band zur späteren Verarbeitung aufgerufen, so sind die gerade im Speicher stehenden Sprites verloren.

Zeilen 14660-14670: Diese Routine schaltet den Sprite aus, normalisiert den Speicher und beendet das Programm.

Zeilen 14680-14730: Mit X können die aktuellen Sprite-Daten mit Daten in einer

anderen Position ausgetauscht werden. Dies ist besonders bei Dreiergruppen von Nutzen.

Zeilen 14740-14830: Mit dieser Routine kann der Benutzer in den Mehr-Farbenmodus der Sprites gehen oder diesen verlassen.

Zeile 14750: Ist der mehrfarbige Modus gesetzt (d.h. das entsprechende Bit in dem Mehr-Farben-Register für die Sprites bei Adresse 53276 ist gesetzt) und wird diese Funktion aufgerufen, so wird der mehrfarbige Modus für Sprite 0 zurückgesetzt (ausgeschaltet).

Zeilen 14760-14800: Die rätselhaften 01, 10 und 11 in diesen Eingabeaufforderungen beziehen sich auf Bitkombinationen in dem Sprite-Raster. Im mehrfarbigen Modus wird davon ausgegangen, daß der Sprite nur über eine Länge von 12 Punkten verfügt (obwohl es in Wirklichkeit doppelt so lang ist). Die Bits werden paarweise von links gelesen und bilden natürlich Paare von 00, 01, 10 oder 11. Jede dieser drei Kombinationen erzeugt eine andere Farbe im mehrfarbigen Modus, wobei 00 die Hintergrundfarbe des Bildschirms ist, die von dem Mehr-Farben-Register der Sprites bei Adresse 53285 festgelegt wird. Die Farbe 10 wird von dem normalen Farb-Register für Sprites festgelegt. Die Farbe 11 wird aus dem Mehr-Farben-Register der Sprites bei Adresse 53286 genommen.

Testen von Modul 3.3.3

Sämtliche in dem Kommentar beschriebenen Funktionen sollten verfügbar sein, nachdem das Modul eingegeben wurde. Wie bei dem entsprechenden Modul in Characters wird empfohlen, jede Funktion bei der Eingabe zu testen.

Zusammenfassung

Denkt man etwas über dieses Programm nach, so wird man feststellen, wie einfach Sprites benutzt werden können, nachdem die Funktionen einiger weniger Speicheradressen verstanden wurden. Das Programm selbst erzeugt eine endlose Folge von Sprites, die zur späteren Benutzung auf einem separaten Band gespeichert werden können. Aufgrund der in dem Programm enthaltenen Techniken ist es sehr einfach, derartige Sprites in selbst erstellten Programmen maximal zu nutzen.

Ein Schritt weiter

1. Das Programm sieht eine weitere Sprite-Funktion nicht vor, d.h. die Funktion zur Erweiterung, mit der Höhe oder Breite des Sprites verdoppelt werden kann

(dieselbe Anzahl von Bytes -nur einfach länger). Diese Funktion kann problemlos hinzugefügt werden, da nur das entsprechende Bit in dem Register bei Adresse 53277 für die horizontale Erweiterung und bei Adresse 53271 für die vertikale Erweiterung gesetzt werden muß. Für dieses Programm ist das richtige Bit das Bit 0.

2. Es wäre gut, wenn nur ein Teil einer Gruppe mit Sprites von dem Band gerufen werden könnte, beispielsweise Sprite für Sprite, um entscheiden zu können, ob dieser Sprite in die aktuelle Gruppe kompiliert werden soll oder nicht. Dies könnte mit einer kleinen Änderung der Laderoutine ermöglicht werden.

3.4 HI-RES

Obwohl die Möglichkeiten der vom Benutzer definierten Zeichen und Sprites nahezu unbeschränkt sind, verfügt der Commodore 64 noch über einen weiteren wichtigen Grafik-Modus, die sogenannte Bit-Mapping (Bit-Raster-Grafik). Dies bedeutet, daß der Benutzer nicht mehr mindestens eines der 1000 Zeichenquadrate auf dem normalen Bildschirm adressieren kann, sondern jedes einzelne Pixel (Bildelement) bzw. jeden einzelnen Punkt auf dem Bildschirm setzen kann. In diesem Modus können Linien und Kurven auf dem Bildschirm gezeichnet werden. Um diese Funktion jedoch maximal nutzen zu können, benötigen Sie die Grafikerweiterung SIMONs BASIC für den C64, mit der Ihnen eine Vielzahl von flexiblen Grafik-Befehlen zur Verfügung stehen.
Um das hier vorgestellte Programm verstehen zu können, sind einige Kenntnisse über den Aufbau des Bit-Map-Bildschirms erforderlich. Der Bildschirm selbst umfaßt 320*200 separate Positionen, d.h. insgesamt 64000. Um jede dieser Positionen separat speichern zu können, sind 8000 Bytes im Speicher erforderlich, d.h. 64000 einzelne Bits. Jede der Standard-Zeichenpositionen erfordert acht Bytes (das 8*8 Raster, das für die vom Benutzer definierten Grafiken benutzt wurde). Wird oben links auf dem Bildschirm begonnen, so werden die ersten acht Bytes (0 bis 7) des Bildschirmspeichers dazu benutzt, die Position zu erstellen, die auf dem normalen Bildschirm die erste Zeichenposition darstellt. Die zweiten acht Bytes bilden das zweite 8*8 Raster usw. über die ganze Zeile. Da eine Zeile über 40 Zeichenpositionen verfügt, enthält jede Zeile 320 Bytes. Da im Bit-Map-Modus einzelne Pixels adressiert werden können, können in dieser Zeile mit 8*8-Rastern acht einzelne Linien in Pixelstärke stehen. (Würden sie jedoch gezeichnet, so würden sie wie ein fester Balken aussehen.)
Der für die Aufnahme des Bit-Map-Bildschirms erforderliche Speicherplatz von 8K kann natürlich nicht in dem normalen 1K-Bildschirmspeicher gespeichert werden und kann auch diesen Bereich nicht als Teil seines Arbeitsbereichs benut-

zen, da die Adressen 1024 bis 2023 für die Speicherung der Farbinformationen für den Bit-Map-Bildschirm benutzt werden. In dem folgenden Programm wurde der Anfang des Bildschirmspeichers auf Adresse 8192 festgelegt, so daß ein Speicherplatz von 6K für das Basic-Programm bleibt, wobei die Möglichkeit besteht, Basic zu verschieben, wenn das Programm entwickelt und verlängert wird. Mit dem folgenden Programm kann der Bit-Map-Bildschirm als Notizblock benutzt werden, wobei entweder die Cursortasten oder ein einfacher Algorithmus für das Zeichnen von Linien benutzt wird, um einen Entwurf auf dem Bildschirm zu erstellen.

Hi-Res: Variablentabelle

DX	Abstand zwischen den Enden der Linie entlang der X-Achse
DY	Abstand zwischen den Enden der Linie entlang der Y-Achse
FN PE	Der Wert, der mit POKE in PP gesetzt werden muß, um Bildelement X, Y zu löschen
FN PP	Die Position des Bytes, in das Pixel X, Y fällt
FN PV	Der Wert, der mit POKE in PP gesetzt werden muß, um Pixel X, Y zu setzen
MO	Der aktuelle Modus des Programms Bildschirmanfang
SL	Die Neigung der zu zeichnenden Linie
X1,X2	X-Koordinaten der Enden der zu zeichnenden Linie
Y1, Y2	Y-Koordinaten der Enden der zu zeichnenden Linie

MODUL 3.4.1

```
10000 REM#*****************************
10010 REM HI-RES BILDSCHIRM INITIALIS.
10020 REM******************************
10022 CL$="":INPUT "■BILDSCHIRM LOESCHE
N (J/N):";CL$
10025 REM POKE 44,64:POKE 43,1:POKE 1638
4,0:CLR
10027 DEF FNPP(X)=SC+320*INT(Y/8)+8*INT(
X/8)+(Y AND 7)
10028 DEF FNPV(X)=PEEK(FNPP(X)) OR (2↑(7
-(X AND 7)))
10029 DEF FNPE(X)=PEEK(FNPP(X)) AND (255
-2↑(7-(X AND 7)))
10030 POKE 53272,(PEEK(53272))OR 8:POKE
```

```
53265,PEEK(53265) OR32:SC=8192
10035 IF CL$="N" THEN 10050
10040 FOR I=SC TO SC+7999:POKE I,0:NEXT
10050 FOR I=1024 TO 2023:POKE I,6*16+12:
NEXT
10060 MO%(0)=2:MO%(1)=5:MO%(2)=10
```

Mit diesem Modul wird der Bildschirmspeicher für den Bit-Map-Modus konfiguriert, werden einige nützliche Funktionen definiert und der hochauflösende Bildschirm gelöscht.

Kommentar

Zeile 10025: Die POKE-Befehle in dieser REM-Anweisung sind zur Ausführung dieses Programms nicht erforderlich. Sie wurden aufgenommen, damit die erforderlichen Informationen zur Verschiebung von Basic vorhanden sind, falls das Programm so erweitert werden soll, daß es im Bildschirmspeicher über die Adresse 8192 hinausgeht. Wie bei dem Sprites-Programm müssen die POKE-Befehle in ein Ladeprogramm aufgenommen werden, das VOR dem Hauptprogramm ausgeführt wird. Für das hier angegebene Programm reicht der Speicherplatz von 6K bis zu der Adresse 8192 längstens aus- so daß es nicht einmal erforderlich ist, eine Grenze für den Anfang von Basic zu setzen.

Zeilen 10027-10029: Die Benutzung dieser Funktionen wird in der Variablentabelle angegeben.

Zeile 10030: 53272 ist das Register, mit dem normalerweise gesteuert wird, an welcher Stelle der VIC II nach Zeichendaten sucht. In diesem Fall legt es den Anfang des Bit-Map-Bildschirms fest. Wird hier mit dem POKE-Befehl 8 angegeben, so wird der Bildschirmanfang auf 8192 festgelegt. Wird 32 mit POKE in Adresse 53265 gesetzt, so wird dadurch der Bit-Map-Modus festgelegt.

Zeilen 10035-10040: Mit Zeile 10022 hatte der Benutzer die Möglichkeit, den Bildschirm zu löschen. Wird das Programm während der Entwicklung gestoppt und werden RUN-RESTORE betätigt, so können Änderungen an dem Programm vorgenommen werden, ohne daß der Bildschirminhalt selbst betroffen ist. Wird das Programm erneut ausgeführt, so wird Zeit gespart, indem die 8000 Bytes nicht gelöscht werden müssen.

Zeile 10050: Mit dieser Zeile wird der normale Bildschirmspeicherbereich gelöscht, in dem nun die Farbdaten für jede der 1000 normalen Zeichenpositionen stehen.

Testen von Modul 3.4.1

Bei der ersten Ausführung des Programms sollten auf dem Bildschirm sofort sinnlose Zeichen angezeigt werden. Sie werden nach und nach gelöscht, wobei auf dem Bildschirm noch immer farbige Quadrate stehen können, die den Zeichenpositionen auf dem Bildschirm im Normalmodus entsprechen. Sie sollten dann ebenfalls langsam gelöscht werden und der Bildschirm sollte weiß sein. Nach Beendigung des Moduls werden die RUN- und RESTORE-Tasten betätigt, um zum Normalmodus zurückzukehren.

MODUL 3.4.2

```
11000 REM#******************************
11010 REM ZEICHNEN
11020 REM#******************************
11030 X=160:Y=96:MO=1:POKE 1024,(PEEK(10
24)AND240) OR (MO*2)
11040 TT=PEEK(FNPP(X))
11042 GET A$:IF A$<>"" THEN 11050
11044 POKE FNPP(X),FNPV(X):POKE FNPP(X),
FNPE(X):GOTO 11042
11050 POKE FNPP(X),TT
11060 IF MO<3 THEN X=X-(A$="■" AND X<319
)+(A$="■" AND X>0)
11062 IF MO=3 THEN X=X-10*(A$="■" AND X<
310)+10*(A$="■" AND X>10)
11070 IF MO<3 THEN Y=Y-(A$="■" AND Y<191
)+(A$="□" AND Y>0)
11072 IF MO=3 THEN Y=Y-10*(A$="■" AND Y<
182)+10*(A$="□" AND Y>10)
11075 IFA$="*"THEN MO=MO+1:MO=MO+4*(MO>3
):POKE1024,(PEEK(1024)AND240)OR(MO*2)
11080 IF MO=1 THEN POKE FNPP(X),FNPV(X)
11090 IF MO=0 THEN POKE FNPP(X),FNPE(X)
11100 IF A$="1" THEN X1=X:Y1=Y
11110 IF A$="2" THEN X2=X:Y2=Y
11120 IF A$="L" THEN GOSUB 12000
11200 GOTO 11040
11499 GOTO 11499
```

Mit diesem Modul kann ein blinkendes Pixel über dem Bildschirm hin- und herbewegt werden, wobei einzelne Pixels angezeigt und gelöscht werden.

Kommentar

Zeile 11030: X und Y sind die Koordinaten des Pixels auf dem 300*200-Bildschirm. Der blinkende Pixel-Cursor wird in die Mitte des Bildschirms gesetzt. In die erste Position im normalen Bildschirmspeicher wird mit POKE ein Wert gesetzt, der einen Farbindikator des aktuellen Modus erzeugt (schwarz 0, rot 1, violett 2, blau 3). Die Auswirkungen der verschiedenen Betriebsarten werden später erläutert.

Zeile 11040: Der Status des Bildschirms in der Position, in der der Cursor aufblinken soll, wird ermittelt.

Zeilen 11042-11050: Der Cursor blinkt immer wieder auf, bis eine Taste betätigt wird.

Zeilen 11060-11072: Im Modus 3 wird durch Betätigung der Cursortaste das blinkende Pixel zehn Stellen in der gewünschten Richtung (innerhalb der Bildschirmgrenzen) bewegt. Im Modus 0, 1 und 2 wird der Cursor immer nur um ein Leerzeichen weiter bewegt.

Zeile 11075: Mit den allein, d.h. ohne die SHIFT-Taste benutzten Funktionstasten von oben
nach unten werden die Betriebsarten festgelegt. Wird der Modus geändert, so wird der Farbindikator geändert.

Zeilen 11080-11090: Ist der Modus Null (schwarz), so wird das Pixel bei der Cursorposition gelöscht. Ist der Modus 1 (rot), so wird das Pixel ausgezeichnet. In den beiden restlichen Betriebsarten kann der Cursor langsam oder schnell hin- und herbewegt werden, ohne daß die Bildschirmanzeige davon betroffen ist.

Zeilen 11100-11120: Diese Eingaben beziehen sich auf das nächste Modul.

Testen von Modul 3.4.2

Der kleine Cursor sollte auf dem Bildschirm hin- und herbewegt werden können, wobei Elemente gezeichnet oder gelöscht werden.

MODUL 3.4.3

```
12000 REM#*****************************
12010 REM ZEICHNEN DER LINIE
12020 REM******************************
12025 X=X1:Y=Y1
12030 DX=X2-X1+SGN(X2-X1):DY=Y2-Y1+SGN(Y
2-Y1)
12032 IF ABS(DY)>ABS(DX) THEN 12200
12035 SL=ABS(DY/DX)-0.5
12040 FOR I=1 TO ABS(DX)
12050 IF MODE=1 THEN POKE FNPP(X),FNPV(X
)
12055 IF MODE=0 THEN POKE FNPP(X),FNPE(X
)
12060 IF SL>0 THEN Y=Y+SGN(DY):SL=SL-1:G
OTO 12060
12070 SL=SL+ABS(DY/DX)
12100 X=X+SGN(DX):NEXT I
12120 RETURN
12200 SL=ABS(DX/DY)-0.5
12210 FOR I=1 TO ABS(DY)
12220 IF MODE=1 THEN POKE FNPP(X),FNPV(X
)
12225 IF MODE=0 THEN POKE FNPP(X),FNPE(X
)
12230 IF SL>0 THEN X=X+SGN(DX):SL=SL-1:G
OTO 12230
12240 SL=SL+ABS(DX/DY)
12250 Y=Y+SGN(DY):NEXT I
12300 RETURN
```

Mit diesem Modul können gerade Linien zwischen vom Benutzer definierten Punkten gezeichnet werden. Hier handelt es sich um eine Anwendung einer als Bresenham's Algorithmus bekannten Methode. Eine Version dieser Methode wird häufig in den Basic-Programmen benutzt, die über Befehle für das Zeichnen von Linien verfügen.

Kommentar

Zeile 12025: Die Werte X1 und Y1 wurden definiert, als der Benutzer 1 eingegeben hat - dort wurden sie der X- bzw. Y-Position des Cursors gleichgesetzt. X2 und Y2 wurden bei der Eingabe von 2 gesetzt. Die Linie wird von X1, Y1 gezeichnet.

Zeile 12030: DX und DY wurden gleich der Entfernung zwischen X1 und X2 bzw. Y1 und Y2 plus Eins gesetzt. Bei der SGN-Funktion spielt es keine Rolle, ob die Distanz positiv oder negativ ist (ist sie negativ, so wird minus 1 und nicht 1 hinzugefügt).

Zeile 12032: Der Algorithmus für das Zeichnen von Linien benutzt die größere der beiden Differenzen als Basis für die Berechnungen, so daß mit zwei separaten Routinen schneller gearbeitet wird.

Zeile 12035: SL ist die Neigung bzw. das Verhältnis zwischen DX und DY minus 0,5.

Zeile 12040: Die Schleife ist so lang wie die Differenz entlang der X-Koordinate.

Zeilen 12050-12055: Je nachdem, ob mit Modus 0 oder 1 gearbeitet wird, wird ein einzelner Punkt auf der Linie gelöscht oder gezeichnet. Im Modus 2 oder 3 geschieht nichts.

Zeile 12060: Je nach Verhältnis zwischen DX und DY kann nun mit SL angegeben werden, daß der nächste Punkt entlang der Y-Achse nach oben oder unten geschoben werden soll. In diesem Fall wird die Y-Position geändert und SL um Eins gekürzt.

Zeile 12070: Der Neigungswert wird jedesmal zu SL hinzugefügt, nachdem ein Punkt ausgedruckt wurde.

Zeile 12100: Die X-Position wird bei jeder Wiederholung der Schleife inkrementiert. Auch hier berücksichtigt die SGN-Funktion Linien, die auf der Achse zurückgehen.

Zeilen 12200-12250: Genau dieselbe Routine für die Fälle, in denen DY größer ist als DX.

Testen von Modul 3.4.3

Hier sollte der Benutzer in der Lage sein, einen Anfangs- und Endpunkt für eine Linie (1 und 2) anzugeben, die Linie zu zeichnen oder eine bestehende Linie zu löschen, je nachdem, ob Modus 1 oder 0 gesetzt ist.

Zusammenfassung

Dieses Programm soll nur das Interesse an den weiteren Möglichkeiten des Bit-Map-Modus wecken. Sollen die Bit-Map-Grafiken voll eingesetzt werden, so muß sorgfältig überlegt werden, welches Ziel erreicht werden soll. Außerdem sind häufig komplexe mathematische Vorgänge erforderlich, um dieses Ziel zu erreichen. Möchte der Benutzer weiter in die Technik des Bit-Mapping einsteigen, so wird diese Aufgabe durch die hier angegebenen Techniken und die Funktionen zur Auffindung der einzelnen Pixels wesentlich einfacher gestaltet.

Ein Schritt weiter

1. Warum soll nicht eine Funktion hinzugefügt werden, mit der ein Bildschirm mit Grafiken auf Band gespeichert werden kann - hierzu ist ein relativ langes Band erforderlich, die Routine ist jedoch recht einfach.
2. Die Bücher über Computergrafiken enthalten eine Reihe von Algorithmen, mit denen Kreise und Bögen gezeichnet werden können. Warum soll nicht an das Ende des Programms ein entsprechendes Modul hinzugefügt werden - der Hauptnachteil ist allerdings eine langsamere Ausführung.

KAPITEL 4

DER C64 ALS SEKRETÄRIN

Früher oder später stellen die meisten Besitzer von Homecomputern fest, daß sich ihr neuer digitaler Freund besonders bewährt, wenn er Informationen speichert, diese verarbeitet und sie auf so vielfältige Weise darstellt, daß dies äußerst arbeitsaufwendig wäre, müßte es manuell geschehen. In diesem Augenblick fangen sie damit an, einfache Programme zu schreiben, mit denen die Namen ihrer Freunde und die Adressen gespeichert werden, oder mit denen beispielsweise eine Plattensammlung festgehalten wird. Schließlich haben sie etwa ein halbes Dutzend Programme, jedes für einen bestimmten Zweck, und dennoch werden für jedes Programm in etwa dieselben Methoden benutzt.
In diesem Kapitel wird ein Abschnitt mit etwas umfangreicheren Programmen begonnen, in dem überprüft wird, wie ein einzelnes Programm geschrieben werden kann, um den verschiedensten Ablageanforderungen gerecht zu werden, ohne daß das Programm ständig neu geschrieben werden muß, wenn eine neue Anwendung benutzt werden soll.

4.1 UNIFILE

Das erste Programm trägt den Namen Unifile. In der hier vorgestellten Form kann es bis zu 500 Einträge speichern, ermöglicht dem Benutzer, diese Einträge nach bestimmten Elementen zu durchsuchen, die Einträge zu ändern und zu löschen. Abgesehen von dem breiten Anwendungsfeld eines derartigen Programms hoffe ich, daß es dem Benutzer nur durch die reine Eingabe und die Darlegung der benutzten Methoden eine Vielzahl von Ideen für weitere Anwendungen vermittelt.

Unifile: Variablentabelle

IN	Flag-Indikator, mit dem angegeben wird, ob das Programm initialisiert wurde
A%(499,X-1)	Zeichnet die Länge einzelner Elemente in jedem Eintrag auf Haupt-Dateimatrix
A$(499)	Enthält die Namen von Elementtypen für jeden Eintrag
B$(X-1)	Flag-Indikator, mit dem festgelegt wird, ob eine Benutzersuche erfolg-
FF	reich ausgeführt wurde
IT	Anzahl von bis jetzt in der Datei stehenden Einträgen
PO	Wird in der Binärsuche dazu benutzt, die Anzahl erforderlicher Suchmuster anzugeben

PP	Pointer zur Anfangsposition des aktuellen auszudruckenden Elements eines Eintrages
R$	Separator zur Benutzung bei der Sicherung von Daten auf Band
S1	Temporärer Such-Pointer für das Suchmodul des Benutzers
SS	Hauptsuch-Pointer in einem binären Suchmodul
T1$	Temporäre Speicherfolge, mit der der neue Eintrag erstellt wird
TI % (20)	Temporärer Speicher für die Länge von Elementen, die in dem neuen Eintrag erstellt werden
X	Enthält die Anzahl von Elementen, die für jede Datei angegeben werden
Z	Indikator für die Nummer der Programmfunktion, die aus dem Hauptmenü aufgerufen wird

MODUL 4.1.1

```
11000 REM#******************************
11010 REM MENUE
11020 REM*******************************
11030 POKE 53281,7:PRINT "
   [REVERS EIN][GRUEN]UNIFILE"
11040 PRINT "[BLAU]MOEGLICHE BEFEHLE:"

11050 PRINT "[ROT]    1)DATENEINGABE"
11060 PRINT "    2)SUCHEN/ANZEIGEN/AENDE
RN"
11070 PRINT "    3)LADEN/SPEICHERN"
11080 PRINT "    4)NEUES FILE ANLEGEN"
11090 PRINT "    5)BEENDEN"
11100 INPUT "[BLAU]BEFEHL:";Z:PRINT ""
;
11110 IF Z>3 OR IN=1 THEN 11140
11120 PRINT "NOCH KEI
N FILE VORHANDEN.":FOR I=1 TO 1000:NEXT
11130 GOTO 11000
11140 ON Z GOSUB 13000,17000,18000,12000
,11150:GOTO 11000
11150 PRINT "[REVERS E
IN][SCHWARZ]UNIFILE BEENDET":END
```

Dieses Modul umfaßt sämtliche Funktionen, die mit dem Programm zur Verfügung stehen. Außerdem gibt es dem Benutzer die Möglichkeit, zwischen den einzelnen Funktionen zu wählen. Als Faustregel gilt, daß jedes komplexe Programm,

das nicht mit einem klar definierten Menü der einzelnen Programmfunktionen beginnt, kein gutes Programm ist. Stimmen Sie hiermit auch jetzt noch nicht überein, so werden Sie dies spätestens dann tun, wenn Sie zu einem komplexen Programm zurückkehren müssen, das einige Wochen lang nicht benutzt wurde, und wenn Sie eine halbe Stunde brauchen, um durch die Programmauflistung zu gehen und sich daran zu erinnern, was das Programm wie tut.

Kommentar

Zeile 11030: Ein typisches Beispiel für die Benutzung der flexiblen Cursor-Kontrollbefehle des Commodore. Die Zeichenfolge löscht den Bildschirm, bewegt den Cursor um eine Stelle nach unten in die Mitte der Zeile, setzt die RVS ON -Eigenschaft und druckt grün aus.

Zeilen 11110-11130: Kein Programm kann erfolgreich ausgeführt werden, wenn die benutzte Matrix nicht erstellt wurde. In diesem Programm ist die Variable IN auf 1 gesetzt, wenn dies geschieht. Ist IN nicht gleich 1, so können aus dem Menü nur die Initialisierung (Erstellen von Matrizes) und der Programmstopp benutzt werden.

Zeile 11140: Für die Benutzer, für die dieser Befehl neu ist, bieten die Befehle ON...GOSUB und ON...GOTO einfache Möglichkeiten, die Listen mit den schwierigen IF...THEN...GOSUB (GOTO) Anweisungen zu beschränken. Der Befehl wählt den Bestimmungsort in der Liste, der mit Z angegeben wird.

In diesem Stadium kann nur getestet werden, ob das Modul eine sauber angeordnete Menüseite darstellt und eine Eingabe akzeptiert. Die einzige Eingabe, die zu keinem Fehlerbericht führt, ist die Eingabe 5 - Programmstopp.

MODUL 4.1.2

```
12000 REM#*****************************
12010 REM STRUKTUR DES FILES
12020 REM******************************
12030 CLR:DIM A$(499):PRINT "███████████
   [REVERS EIN][ROT]FILE STRUKTUR":IN=1:
R$=CHR$(13)
12035 INPUT "█[SCHWARZ]LADEN SIE VON KAS
SETTE (J/N):";Q$:IF Q$="J" THEN 11000
12040 INPUT "██[BLAU]WIEVIELE FELDER PRO
 DATENSATZ:";X:DIM B$(X-1),A%(499,X-1)
12050 PRINT "█";:FOR I=0 TO X-1:PRINT "[
```

```
PURPUR]NAME VON FELD";STR$(I+1);":";:INP
UTQ$
12060 B$(I)=Q$:NEXT I:GOTO 11000
```

Dieses Modul führt die wichtige Funktion des Erstellens der Felder aus, die für die Speicherung der Programmdaten benutzt werden. Erst wenn dieses Modul aufgerufen wurde, kann das Programm benutzt werden. Nachdem Daten eingegeben wurden, führt das erneute Aufrufen des Moduls zu einem Verlust sämtlicher Daten - der Speicher wird für die Aufnahme einer neuen Datengruppe gelöscht. Die Benutzung der Hauptvariablen wird in der Variablentabelle und in den folgenden Kommentaren zu dem Programm erläutert.

Kommentar

Zeile 12030: Bevor ein Feld dimensioniert wird, muß der Speicher unbedingt gelöscht werden. Geschieht dies nicht, so wird die Fehlermeldung REDIMMED ARRAY angezeigt.

Zeile 12040: Unifile schreibt dem Benutzer nicht vor, wie viele Elemente ein typischer Eintrag enthalten muß. Dies kann der Benutzer selbst festlegen. Nachdem dies geschehen ist, konfiguriert das Programm das Zeigerfeld A% und das Feld für den Elementtitel B$ dementsprechend.

Zeilen 12050-12060: Nachdem die Anzahl von Elementen pro Eintrag angegeben wurde, werden die Elemente benannt, z.B. Name, Adresse, Telefonnummer. Da der Speicher gelöscht wurde, kann das Modul nicht zu dem Menü zurückkehren. Hier muß die Zeile angegeben werden, in die mit GOTO gegangen werden soll.

Testen von Modul 4.1.2

Beim Aufruf des Moduls wird der Benutzer gebeten, die Anzahl von Elementen pro Datei anzugeben und den Elementen Namen zuzuweisen.

MODUL 4.1.3

```
13000 REM#********************************
13010 REM EINGABE NEUER PUNKTE
13020 REM*********************************
13030 T1$="":PRINT "██████████████████[REV
ERS EIN][ROT]NEUE FELDER
13040 PRINT "████████████████████[GRUEN]";
```

```
IT;" FELDER BIS JETZT."
13050 PRINT "■[BLAU]MOEGLICHE BEFEHLE:"
13060 PRINT "■[SCHWARZ]>[PURPUR]GEBEN SI
E DAS GEWUENSCHTEN FELD EIN"
13070 PRINT "[SCHWARZ]>[PURPUR]ODER 'ZZZ
' UM INS MENUE ZURUECKZUKEHREN"
13080 FOR I=0 TO X-1:PRINT B$(I);":";:IN
PUT Q$:IF Q$="ZZZ" THEN RETURN
13090 IFLEN(T1$)+LEN(Q$)<=255THEN 13110
13100 PRINT "[ROT]EINGABE ZU LANG.":FOR
J=1 TO 3000:NEXT J:RETURN
13110 T1$=T1$+Q$:TI%(I)=LEN(T1$):NEXT I:
PRINT "■[SCHWARZ]BITTE WARTEN !"
13120 GOSUB14000:GOSUB 15000:GOTO 13000
```

Dieses Modul akzeptiert die Eingabe der vom Benutzer angegebenen Elemente und verwandelt sie in einen Eintrag für die Hauptdatei.

Kommentar

Zeile 13080: Mit der Variablen X wird die Anzahl von Wiederholungen festgelegt. Das Programm fordert den Benutzer auf, jedes der angegebenen Elemente einzugeben.

Zeilen 13090-13100: Einzelne Einträge können eine maximale Länge von 255 Zeichen aufweisen - die Maximallänge einer einzelnen Zeichenfolge bei dem Commodore 64. Mit diesen Zeilen wird geprüft, ob dieser Grenzwert nicht überschritten wird.

Zeile 13110: Das eingegebene Element wird zu der temporären Speicherfolge T1$ hinzugefugt. Die Länge des bis jetzt eingegebenen Eintrags wird in TI% aufgezeichnet. Hier wird darauf hingewiesen, daß TI% in dem Initialisierungsmodul nicht deklariert wurde. Durch Angabe dieser Speicherfolge im Laufe des Programms wird sie automatisch auf zehn Elemente (0 bis 9) dimensioniert. Werden Einträge mit mehr als zehn Elementen gewünscht, so muß eine größere TI in dem Initialisierungsmodul deklariert werden.

Testen von Modul 4.1.3

Werden in diesem Stadium temporäre RETURN-Befehle in den Zeilen 14000 und 15000 eingegeben, so sollte dieses Modul aufgerufen werden können. Der Be-

nutzer wird aufgefordert, Elemente unter den angegebenen Namen einzugeben. Bis jetzt ist noch nicht vorgesehen, diese Elemente in die Datei einzugeben.

MODUL 4.1.4

```
14000 REM#*****************************
14010 REM SUCHROUTINE
14020 REM******************************
14030 IF IT=0 THEN SS=0:RETURN
14040 PO=INT(LOG(IT)/LOG(2)):SS=2↑PO-1
14050 FOR I=PO TO 0 STEP-1
14060 IF A$(SS)<T1$ THEN SS=SS+2↑I
14070 IF A$(SS)>T1$ THEN SS=SS-2↑I
14080 IF SS<0 THEN SS=0
14090 IF SS>IT-1 THEN SS=IT-1
14100 NEXT I:IF A$(SS)<T1$ THEN SS=SS+1
14110 RETURN
```

Von sämtlichen Moduln in diesem Programm sieht dieses wahrscheinlich auf den ersten Blick am schwierigsten aus. In Wirklichkeit ist es sehr einfach, zuerst müssen jedoch die grundlegenden Prinzipien verstanden werden, die hinter einer Suchmethode liegen, die als binäre Suche bezeichnet wird. Mit ihr wird der Arbeitsaufwand drastisch verkürzt, der erforderlich ist, um die richtige Stelle für ein neues Element in einer geordneten Datenliste zu finden.

Hier soll von folgendem Beispiel ausgegangen werden:
Es wurde eine Datei mit 2000 Namen in alphabetischer Reihenfolge erstellt. Nun muß ein neuer Name eingefügt werden, dessen rechtmäßiger Platz bei Position 1731 liegt, obwohl dies erst noch festgelegt werden muß. Die Suchroutine überprüft also zuerst den ersten Namen in der Datei, entscheidet, daß der neue Namen nach diesem ersten Namen angegeben werden muß und geht zu dem zweiten Namen. Nachdem 1732 Namen geprüft wurden, hat die Suchroutine einen Namen gefunden, vor dem der neue Namen stehen soll. Nun weiß sie, daß sie den richtigen Platz für die Einfügung des neuen Namens gefunden hat. Hier handelt es sich um ein sehr direktes Verfahren, das einfach zu programmieren ist. Dies soll jedoch nun mit folgendem Verfahren verglichen werden:
Das Suchverfahren beginnt mit der Überprüfung des Namens in Position 1024 der Datei, da 1024 die größte Zweierpotenz ist, die in die Gesamtanzahl von Namen in der Datei gesetzt werden kann. Der Name bei 1024 ist alphabetisch kleiner als der neue Name. Deshalb fügt die Suchroutine 1024/2 zu der ursprünglichen Adresse 1024 hinzu und geht zu dem Namen mit der Nummer 1536. Dieser Name ist noch

immer kleiner als der neue Name, so daß 1024/4 zu 1536 hinzugefügt wird, was 1792 ergibt. Nun geschieht etwas anderes - der Name mit der Nummer 1792 ist alphabetisch größer als der neue Name. Dieses Problem wird gelöst, indem 1204/8 subtrahiert wird, was zu einem Wert 1664 führt. Die Suchroutine addiert oder subtrahiert weiterhin abnehmende Zweierpotenz, um ein Suchmuster zu erstellen, das folgendermaßen aussieht:

1644 (danach wird 64 addiert)
1728 (danach wird 32 addiert)
1760 (danach wird 16 subtrahiert)
1744 (danach wird 8 subtrahiert)
1736 (danach wird 4 subtrahiert)
1732 (danach wird 2 subtrahiert)
1730 (danach wird 1 addiert)

Die Anzahl von Vergleichen, die erforderlich sind, um die richtige Stelle in der Datei zu finden, wurde von 1732 auf 10 gekürzt. Dies ist ein eindeutiger Beweis für die Leistungsfähigkeit der binären Suche.

Zeile 14030: Stehen in der Datei noch keine Elemente, so muß die richtige Position nicht berechnet werden.

Zeile 14040: Mit der LOG-Funktion wird die höchste Zweierpotenz ermittelt, die der aktuellen Anzahl von Elementen entspricht.

Zeilen 14050-14100: Der Binärsprung wird ausgeführ wobei geprüft wird, ob die Suche nicht über das Ende der Datei hinausgeht. Ein letzter Vergleich wird vorgenommen, wenn die Schleife beendet ist und die richtige Position ermittelt und in der Variablen SS gespeichert wurde.

Testen von Modul 4.1.4

Dieses Modul kann erst vollständig getestet werden, nachdem das nächste Modul eingegeben wurde. Allerdings kann geprüft werden, ob die Syntax korrekt ist, indem einfach das Einfüge-Modul aufgerufen wird, aus dem dieses Modul aufgerufen wird.

MODUL 4.1.5

```
15000 REM#********************************
15010 REM EINFUEGEN
15020 REM*********************************
15030 IF IT=0 THEN GOTO 15060
```

```
15040 FOR I=IT TO SS+1 STEP -1:A$(I)=A$(
I-1)
15050 FOR J=0 TO X-1:A%(I,J)=A%(I-1,J):N
EXT J,I
15060 A$(SS)=T1$:FOR I=0 TO X-1:A%(SS,I)
=TI%(I):NEXT:IT=IT+1:RETURN
```

Nachdem die richtige Position festgelegt wurde, verschiebt dieses Modul sämtliche Einträge um eine Stelle weiter in der Datei, zusammen mit den zugeordneten Pointern in A%. Der neue Eintrag wird in Position SS der Datei gesetzt. Die Pointer, die die Länge der einzelnen Elemente angeben, werden in dieselbe Position in A% gesetzt.

Testen der Moduln 4.1.4 und 4.1.5

Nun sollte der Benutzer in der Lage sein, Einträge in die Datei einzugeben, die in alphabetischer Reihenfolge angeordnet werden. Um dies zu prüfen, muß das Programm gestoppt und der Inhalt von A$(0), A$(1) usw. im direkten Modus ausgedruckt werden. Außerdem muß geprüft werden, ob die in derselben Zeile von A% gespeicherten Zeiger auch wirklich auf das letzte Zeichen jedes Elements in dem Eintrag zeigen.

MODUL 4.1.6

```
18000 REM#******************************
18010 REM LADEN/SPEICHERN
18020 REM*******************************
18030 PRINT "■[BLAU]KASSETTE EINLEGEN, D
ANN [REVERS EIN]RETURNR[REVERS AUS]--"
18040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
18050 PRINT "■[ROT]MOEGLICHE BEFEHLE:":P
RINT "■[GRUEN] 1)DATEN SPEICHERN
18055 PRINT "2)DATEN LADEN"
18060 INPUT "■[BLAU]BEFEHL:";Q:ON Q GOTO
 18070,18120:RETURN
18070 POKE 1,7:FOR I=1 TO 2000:NEXT
18080 OPEN 1,1,1,"UNIFILE":PRINT#1,IT,R$
,X
18090 FOR I=0 TO IT-1:PRINT#1,A$(I):FOR
J=0 TO X-1:PRINT#1,A%(I,J):NEXT J,I
```

```
18100 FOR I=0 TO X-1:PRINT#1,B$(I):NEXT
18110 CLOSE1:RETURN
18120 OPEN 1,1,0,"UNIFILE":INPUT#1,IT,X:
DIM B$(X-1),A%(499,X-1)
18130 FOR I=0 TO IT-1
18132 GET#1,T$:IF T$<>CHR$(13) THEN A$(I
)=A$(I)+T$:GOTO 18132
18134 FOR J=0 TO X-1:INPUT#1,A%(I,J):NEX
T J,I
18140 FOR I=0 TO X-1:INPUT#1,B$(I):NEXT
18150 CLOSE1:RETURN
```

Nachdem nun einige Daten in die Datei eingegeben werden können, müssen als erstes einige Daten auf Band gespeichert werden. Müssen neue Moduln eingegeben oder Zeilen geändert werden, um Fehler zu korrigieren, so müssen dann nicht sämtliche Daten erneut eingegeben werden.

Kommentar

Zeile 18040: Nachdem das Band eingelegt wurde, wird es als erstes in den RECORD- oder PLAY-Modus versetzt und genau bis zu der Stelle vorgespult, die von dem Bandzähler angegeben wird. Ist diese Stelle erreicht, so wird der Motor des Kassettenrecorders mit Hilfe dieser beiden POKE-Befehle bei Betätigung der RETURN-Taste ausgeschaltet.

Zeile 18070: Der Motor des Recorders wird wieder eingeschaltet, bevor Daten aufgezeichnet und zusätzlich zu dem automatisch vom Betriebssystem hinzugefügten Vorsatz Anfangsinformationen ausgedruckt werden. Dadurch wird gewährleistet, daß keine Aufzeichnung auf dem nichtmagnetischen Bandvorspann vorgenommen wird, wenn am Bandanfang begonnen wird.

Zeilen 18120-18140: Nun werden Daten, die in die Datei gedruckt wurden, wieder aufgerufen. Da die Zeichenfolgen in der Hauptdatei eine Länge von mehr als 80 Zeichen aufweisen können, kann der INPUT-Befehl nicht benutzt werden. Statt dessen wird jedes Zeichen der Zeichenfolgen in der Hauptdatei separat mit GET geholt. Jeder Eintrag wird als vollständig betrachtet, wenn vom Band ein Zeilenschaltungs-Zeichen geholt wird.

Testen von Modul 4.1.6

Bei diesem Modul wird einfach getestet, ob Daten in das Programm eingegeben, auf Band gespeichert und dann erneut geladen werden können.

MODUL 4.1.7

```
17000 REM#*****************************
17010 REM SUCHEN
17020 REM******************************
17030 S1=0:FF=0:PRINT "
[ROT][REVERS EIN]SEARCH"
17040 PRINT "[BLAU]MOEGLICHE BEFEHLE:"
17050 PRINT " [SCHWARZ]>[GRUEN]EINGABE
EINES SUCHSTRINGS FUER NORMALE  SUCHE
17060 PRINT " [SCHWARZ]>[GRUEN]SETZEN SI
E 'III' DAVOR FUER INITIALSUCHE"
17070 PRINT " [SCHWARZ]>[GRUEN]SETZEN SI
E 'SSS' DAVOR FUER SPEZIELLE SUCHE"
17080 PRINT " [SCHWARZ]>[GRUEN]RETURN[RE
VERS AUS] FUER DAS ERSTE FELD DES FILES
17090 PRINT "[PURPUR]●●●●●●●●●●●●●●●●●
●●●●●●●●●●●●●●●●●●●●●●●";
17100 T1$="":INPUT"[BLAU]SUCHBEFEHL:";
T1$
17110 IF LEFT$(T1$,3)<>"III" THEN 17140
17120 T1$=RIGHT$(T1$,LEN(T1$)-3):GOSUB 1
4000:S1=SS:IF S1>IT-1 THEN RETURN
17130 GOTO 17240
17140 IF LEFT$(T1$,3)<>"SSS" THEN 17190
17150 FF=0:T1$=RIGHT$(T1$,LEN(T1$)-3):FO
R I=S1 TO IT-1:FOR J=1 TO LEN(A$(I))
17160 IF MID$(A$(I),J,LEN(T1$))=T1$ THEN
 FF=1:S1=I:J=LEN(A$(I)):I=IT-1
17170 NEXT J,I:IF FF=1 THEN T1$="SSS"+T1
$:GOTO 17240
17180 RETURN
17190 IF T1$="" THEN 17240
17200 FF=0:FOR I=S1 TO IT-1:PP=0:FOR J=0
 TO X-1
17210 IF MID$(A$(I),PP+1,A%(I,J)-PP)=T1$
THEN FF=1:S1=I:J=X-1:I=IT-1
17220 PP=A%(I,J):NEXT J:NEXT I:IF FF=1 T
HEN 17240
17230 RETURN
```

```
17240 IF S1>IT-1 THEN S1=IT-1
17250 IF IT=0 THEN RETURN
17260 IF S1<0 THEN S1=0
17270 PRINT "[ROT]DATENSATZ";S1+1;":-
":PP=0
17280 FOR I=0 TO X-1:PRINT"[GRUEN]";B$(I
);":[BLAU]";MID$(A$(S1),PP+1,A%(S1,I)-PP
)
17290 PP=A%(S1,I):NEXT I:S1=S1+1:PRINT "
            "
17300 PRINT "MOEGLICHE [REVERS EIN][ROT]
SUCH[REVERS AUS][BLAU]BEFEHLE:"
17310 PRINT " [SCHWARZ]>[PURPUR][REVERS
 EIN]RETURN[REVERS AUS] UM ZUM NAECHSTEN
 FELD UEBERZU-   GEHEN
17320 PRINT " [SCHWARZ]>[PURPUR]'AAA' FU
ER AENDERUNG
17330 PRINT " [SCHWARZ]>[PURPUR]'CCC' UM
 DIE SUCHE FORTZUFUEHREN
17340 PRINT " [SCHWARZ]>[PURPUR]'#' + NU
MMER UM DEN ZAEHLER ZU ERHOE-        HE
N
17350 PRINT " [SCHWARZ]>[PURPUR]'ZZZ' UM
 DIE FUNKTION ABZUBRECHEN
17360 P$="":INPUT "[BLAU]BEFEHL:";P$
17370 IF P$="CCC" THEN 17110
17380 IF P$="" THEN 17240
17390 IF P$="AAA" THEN GOSUB 16000:GOTO
17240
17400 IF P$="ZZZ" THEN RETURN
17410 IF LEFT$(P$,1)="#" THEN S1=S1+VAL(
MID$(P$,2))-1:GOTO 17240
17420 S1=S1-1:GOTO 17240
```

Nachdem die Daten in den C64 gesetzt wurden, möchte man sie wieder aufrufen. Genau dies ist die Aufgabe dieses Moduls, d.h. die Rückübertragung von Informationen, die auf verschiedene Art und Weise gespeichert wurden, vom das Ablagesystem besonders nützlich zu gestalten.

Kommentar

Zeilen 17110-17130: Stehen vor dem zu suchenden Element die Buchstaben III, so wird das binäre Suchmodul aufgerufen, um einen Eintrag zu suchen, der mit den angegebenen Buchstaben beginnt, oder den Eintrag, der diesem Eintrag am nächsten kommt, kein entsprechender Eintrag gefunden wird. Hier handelt es sich nicht unbedingt um das erste Element in der Datei, das der Bedingung gerecht wird. Wird die ursprüngliche Suchfunktion dazu benutzt, beispielsweise den ersten mit L beginnenden Eintrag zu suchen, so erhält man einen Eintrag, der mit L beginnt und kann dann zurückblättern, um zu sehen, ob es sich um den ersten Eintrag handelt. Durch IIILA kommt man näher, während IIILAAAA die Lösung bringt, wenn keine äußerst ungewöhnliche Namen gespeichert wurden.

Zeilen 17140-17180: Steht vor dem zu suchenden Element SSS, so wird die ganze Datei nach dieser Zeichenkombination durchsucht - hier braucht es sich nicht um ein ganzes Element zu handeln. Durch Angabe von SSSLO würden sämtliche Einträge gefunden, die London, Lotte oder Hallo enthalten. Diese Suchmethode ist selbstverständlich langsamer als die anderen.

Zeile 17190: Wurde die RETURN-Taste ohne eine Eingabe betätigt, so wird das erste Element in der Datei angezeigt.

Zeilen 17200-17230: Jede andere Eingabe wird als ein vollständiges Element betrachtet, nach dem gesucht werden muß. Nur die Einträge, die ein Element in genau dieser Form enthalten, werden zurückgegeben. In dieser Routine muß beachtet werden, wie das Pointer-Feld A%, die die Position des letzten Zeichens jedes Elements in einem Eintrag angibt, zur Ausgabe von Elementen aus dem Eintrag benutzt wird, obwohl keine sichtbare Marke für die Elemente vorhanden ist, wenn ein Eintrag im direkten Modus ausgedruckt wird.

Zeilen 17270-17290: Der Eintrag, den das Suchmodul gefunden hat, wird auf dem Bildschirm ausgedruckt.

Zeilen 17300-17420: Nachdem ein Eintrag angezeigt wurde, bietet das Programm nun die Möglichkeit, den nächsten Eintrag anzuzeigen, den Eintrag zu ändern, mit der angegebenen Suche fortzufahren, durch Eingabe von NN zu einem anderen Eintrag zu gehen, wobei NN eine positive oder negative Zahl darstellt, um die in der Datei vor- oder zurückgegangen werden muß, oder zu dem Hauptmenü zurückzukehren. Wird keine erkennbare Eingabe vorgenommen, so wird derselbe Eintrag erneut angezeigt.

Testen von Modul 4.1.7

Nun sollte der Benutzer in der Lage sein, beliebige gespeicherte Daten anzuzeigen und die Daten mit den beschriebenen Suchmethoden zu durchsuchen. Bis jetzt können noch keine Einträge geändert werden.

MODUL 4.1.8

```
16000 REM#******************************
16010 REM EINGABE AENDERN
16020 REM*******************************
16030 S1=S1-1:T1$=""
16040 PP=0:FOR I=0 TO X-1:PRINT "[ROT]
DATENSATZ ";S1+1;":-"
16050 PRINT"[GRUEN]";B$(I);":[BLAU]";MID
$(A$(S1),PP+1,A%(S1,I)-PP)
16060 PRINT "
[ROT][REVERS EIN]AMEND"
16070 PRINT "[BLAU]MOEGLICHE BEFEHLE:"
16080 PRINT " [SCHWARZ]>[GRUEN][REVERS
EIN]RETURN[REVERS AUS] LAESST DEN DATENS
ATZ UNGEAEN-    DERT"
16090 PRINT " [SCHWARZ]>[GRUEN]GEBEN SIE
 EIN NEUES FELD EIN UM "
16095 PRINT "  EIN ANDERES ZU ERSETZEN"
16100 PRINT " [SCHWARZ]>[GRUEN]'DDD' LOE
SCHT EINEN GANZEN EIINTRAG"
16110 PRINT " [SCHWARZ]>[GRUEN]'ZZZ' LAE
SST DEN DATENSATZ UNGEAENDERT"
16120 Q$="":INPUT "[BLAU]BEFEHL:";Q$
16130 IF Q$="ZZZ" THEN RETURN
16140 IF Q$="" THEN Q$=MID$(A$(S1),PP+1,
A%(S1,I)-PP)
16150 IFQ$="DDD"THEN GOSUB 16180:RETURN
16160 PP=A%(S1,I):T1$=T1$+Q$:TI%(I)=LEN(
T1$):NEXT I:GOSUB 16180:GOSUB 14000
16170 S1=SS:GOSUB 15000:RETURN
16180 FOR J=S1 TO IT-1:A$(J)=A$(J+1):FOR
 K=0 TO X-1:A%(J,K)=A%(J+1,K):NEXT K,J
16190 IT=IT-1:RETURN
```

Mit diesem Modul sollen Änderungen an Elementen in Einträgen vorgenommen werden, die schon gespeichert wurden, ohne daß der ganze Eintrag neu eingegeben zu werden braucht. Außerdem sollen Elemente oder ganze Einträge gelöscht werden können.

Kommentar

Zeile 16140: Das Modul arbeitet wie das Haupteingabemodul. Wird jedoch die RETURN-Taste betätigt, so wird das eingegebene Element als das aktuelle Element auf dem Bildschirm definiert.

Zeile 16180: Mit dieser Routine werden alle folgenden Einträge in der Datei um eine Stelle nach unten bewegt, wobei der aktuelle Eintrag gelöscht wird.

Testen von Modul 4.1.8

Nun sollte der Benutzer in der Lage sein, Elemente in einem Eintrag zu ändern, der in dem Suchmodul steht, oder den ganzen Eintrag zu löschen. Wird dieses Modul richtig ausgeführt, so ist das Programm einsatzbereit.

Zusammenfassung

Damit wurde die Eingabe eines wichtigen und komplexen Programms beendet, das hoffentlich bei vielen Anwendungen von Nutzen sein wird. Im Laufe dieses Verfahrens wurden auch eine Reihe von Techniken erläutert, die hilfreich sein werden, wenn der Benutzer in eigenen, anspruchsvollen Programmen nicht numerische Daten speichern und verarbeiten möchte.
Hat sich der Benutzer jedoch die Mühe gegeben, die vorgenommenen Eingaben in Gedanken nachzuvollziehen, die Funktionen der einzelnen Zeilen, sowie die Gesamtfunktionen der Moduln zu überprüfen, so weiß er nun, daß umfangreiche und komplexe Programme nicht immer so schwierig sind, wie sie auszusehen scheinen. Mit Hilfe einer modularen Lösung, bei der das Programm in eine Reihe von überschaubaren Aufgaben unterteilt wird, können Anwendungen wie diese von jedem Benutzer entwickelt werden, der bereit ist, etwas Zeit (und ein klein wenig Mühe) zu investieren.

Ein Schritt weiter

1. Verfügt der Benutzer über einen Drucker, so wird er die Einträge oder Gruppen von Einträgen auf Papier ausgeben wollen. Hierzu wird am einfachsten ein weiterer Befehl zum zweiten Teil des Such-Moduls hinzugefügt.
2. Weiterhin wäre es interessant, zu sehen, ob das Programm neben den nicht-

numerischen Daten auch numerische Daten verarbeiten kann. Zu diesem Zweck müßte ein numerisches Feld mit 500 Elementen erstellt werden, in die Werte und vielleicht einige Suchbefehle wie 'Suche nach Einträgen mit einem Wert, der größer ist als X'; eingegeben werden können. Es gibt eine ganze Reihe von Anwendungen, bei denen die Speicherung eines oder mehrerer numerischer Elemente von Vorteil wäre.

4.2 UNIFILE II DATENBANK

Nachdem Unifile eingegeben und ausgetestet wurde, steht Ihnen vielleicht nicht unbedingt der Sinn nach einer Variation desselben Themas. Ist dies der Fall, so kann dieses Programm für den Augenblick übersprungen und zu erfreulicheren Dingen übergegangen werden. Zu irgendeinem Zeitpunkt werden Sie jedoch zu diesem Programm zurückkehren, um zumindest einige der Probleme zu lösen, mit denen Unifile nicht fertig wird. Unifile ist für Dateien mit einer regelmäßigen Struktur geschaffen, und davon gibt es viele. Außerdem gibt es eine große Anzahl von Anwendungen, bei denen Sie einfach nicht im voraus wissen, wie viele Elemente in einem bestimmten Eintrag stehen. So möchte man vielleicht eine Bibliothek katalogisieren. Dazu könnte das ursprüngliche Unifile-Programm so erstellt werden, daß Autor und Titel angefordert werden. Sobald jedoch mehr als ein Buch pro Autor vorhanden ist, ist die Zuweisung des Autorennamens zu jedem einzelnen Titel sehr platzaufwendig.
Unifile II ist für solche weniger strukturierten Dateien geschaffen. Es ist insofern flexibler als Unifile, als weiter beliebig Elemente zu einem Eintrag hinzugefügt werden können, solange innerhalb der Gesamtgrenze von 255 Zeichen geblieben wird. Außerdem kann eine wesentlich komplexere Suchform angegeben werden, bei der sämtliche Einträge gefunden werden, die bis zu zehn separate Suchziele enthalten. Diese Flexibilität hat jedoch einen Preis, insofern als das Programm komplizierter in der Benutzung ist. Hier gibt es keine der einfachen Eingabeaufforderungen, mit denen festgelegt wird, welches Element als nächstes eingegeben werden muß. Sollen darüber hinaus Elemente innerhalb eines Eintrags mit einem Titel versehen werden, so muß angegeben werden, um welche Titel es sich handelt. Sie müssen dann in einer codierten Form angefügt werden - das Programm weiß nicht, was als nächstes kommt, dies muß der Benutzer wissen.
Da das Programm in der Struktur Unifile ähnlich ist, wird es am einfachsten eingegeben, indem als erstes Unifile selbst geladen wird. Bei der Eingabe von Unifile II werden Sie dann feststellen, daß viele der Programmzeilen identisch oder nahezu identisch sind, selbst wenn sie anders numeriert sind. Werden diese Zeilen neu numeriert, bevor man sich mit den Unterschieden befaßt, so wird sehr viel Zeit gespart.

Unifile II: Variablentabelle

B$(49)	Enthält die angegebenen wahlweisen Elementtitel
EX	Temporärer Indikator, mit dem angegeben wird, daß in dem Änderungsmodul ein zusätzliches Element zu einem Eintrag hinzugefügt wurde
FNA(S1)	Funktion, die aus dem Wert des letzten Zeichens in einem Eintrag die Anzahl von Elementen innerhalb dieses Eintrags ausgibt
FNB(S1)	Funktion, die die Position des letzten Zeichens eines Elementes innerhalb eines Eintrags ermittelt. Diese Funktion muß innerhalb einer Schleife mit einer Schleifenvariablen I benutzt werden, die die Nummer des Elementes angibt
MN	Temporäre Variable, in der die Anzahl von Elementen innerhalb eines eingegebenen Eintrags festgehalten wird
SS$	Aus dem Eintrag auf der Grundlage von FNA und FNB ausgegebenes Element
S2	Temporärer Pointer, der während der Suche benutzt wird
S3	Temporäre Aufzeichnung des Wertes von S1 während der Mehrfachsuche
TI%(49)	Wird temporär zur Speicherung der Position von Elementen innerhalb eines eingegebenen Eintrags benutzt
TN	Die Typ-Nummer eines Elementes, wenn eines angegeben wird

MODUL 4.2.1

```
11000 REM#*******************************
11010 REM MENUE
11020 REM********************************
11030 POKE 53281,7:PRINT "
   [REVERS EIN][GRUEN]UNIFILE"
11040 PRINT"[BLAU]MOEGLICHE BEFEHLE:"
11050 PRINT"[ROT]    1)DATEN EINGEBEN"
11060 PRINT"    2)SUCHEN/ANZEIGEN/AENDER
N"
11070 PRINT"    3)NEUE FELDER ERSTELLEN"

11080 PRINT"    4)LADEN/SPEICHERN"
11090 PRINT"    5)NEUES FILE ANLEGEN"
11100 PRINT"    6)BEENDEN"
11110 INPUT"[BLAU]BEFEHL:";Z:PRINT"";
11120 IF Z>4 OR IN=1 THEN 11150
11130 PRINT"NOCH KEIN FI
```

```
LE VORHANDEN.":FORI=1TO1000:NEXT
11140 GOTO 11000
11150 ON Z GOSUB 13000,17000,19000,20000
,12000,11160:GOTO 11000
11160 PRINT"███████████████████[REVERS E
IN][SCHWARZ]PROGRAMM BEENDET":END
```

Ein Standard-Menümodul.

MODUL 4.2.2

```
12000 REM#*****************************
12010 REM FILE ANLEGEN
12020 REM******************************
12030 CLR:DIM A$(499),B$(49),TI%(49):IN=
1
12040 DEF FNA(S1)=ASC(RIGHT$(A$(S1),1))+
1
12050 DEF FNB(S1)=ASC(RIGHT$(A$(S1),FNA(
S1)-I+1))
12060 GOTO 11000
```

Das Modul initialisiert die Felder und kehrt sofort zu dem Menü zurück.

MODUL 4.2.3

```
13000 REM#*****************************
13010 REM DATEN EINGEBEN
13020 REM******************************
13030 T1$="":NN=-1:TN=0:PRINT"███████████
      [REVERS EIN][ROT]EINGABE"
13040 PRINT"███████████████[GRUEN]";IT;"
DATENSAETZE BIS JETZT"
13050 PRINT"█[BLAU]MOEGLICHE BEFEHLE:"
13060 PRINT"█[SCHWARZ]>[PURPUR]EINGABE D
ES FELDES:"
13070 PRINT"[SCHWARZ]>[PURPUR]'*' UM DEN
 DATENSATZ ZU BEENDEN"
13080 PRINT"[SCHWARZ]>[PURPUR]'↑NN' UM D
AS FELD NN EINZUGEBEN"
13090 PRINT"[SCHWARZ]>[PURPUR]'ZZZ' UM I
```

```
NS MENUE ZURUECKZUKEHREN"
13100 INPUT Q$:IF Q$="ZZZ" THEN RETURN
13110 IF Q$="ZZZ" THEN RETURN
13120 IF LEFT$(Q$,1)<>"↑" THEN 13150
13130 TN=VAL(MID$(Q$,2)):TN=TN+10
13140 PRINT"□";B$(TN-11);":";:GOTO13100
13150 IF TN<>0 THEN Q$=Q$+"↑"+MID$(STR$(
TN),2):TN=0
13160 IF LEN(T1$)+LEN(Q$)+NN+2<=255 THEN
 13180
13170 PRINT"[ROT]DATENSATZ ZU LANG.":FOR
J=1TO3000:NEXTJ:RETURN
13180 IF Q$="*" THEN GOTO 13200
13190 T1$=T1$+Q$:TI%(NN+1)=LEN(T1$):NN=N
N+1:GOTO 13100
13200 FOR I=0 TO NN:T1$=T1$+CHR$(TI%(I))
:NEXT:T1$=T1$+CHR$(NN+1)
13210 PRINT "▣[SCHWARZ]WAIT":GOSUB14000:
GOSUB 15000:GOTO 13000
```

Dieses Modul entspricht dem Eingabemodul von Unifile, ist jedoch aus zwei Gründen komplexer:

1. Innerhalb des Moduls dem Programm mitgeteilt werden, wann ein Eintrag beendet ist. Dies geschieht durch Eingabe eines Sternchens ohne weitere Eingabe.
2. Da die Datei keine regelmäßige Struktur aufweist, können keine regelmäßigen Eingabeaufforderungen für die Namen der einzugebenden Elemente vorgesehen werden. Das Programm sieht jedoch die Benennung von Elementen vor. Diese Namen und die ihnen zuzuweisenden Zahlen werden durch ein nachfolgendes Modul definiert - in diesem Modul kann der Elementname an ein Element angefügt werden, indem als erstes das '↑ ' Symbol gefolgt von der Nummer eingegeben wird, die dem gewünschten Typnamen vorher zugewiesen wurde.

Kommentar

Zeilen 13100-13140: Beginnt ein Eintrag mit dem '↑ ' Symbol, so werden die darauf folgenden Zeichen als die Nummer eines Elementnamens angesehen, der zu dem gerade eingegebenen Element hinzugefügt werden soll. Die Eingabeaufforderung wird nun unter diesem Typnamen wiederholt. Die Typnummer wird am Ende des Elements gespeichert, wobei ihr Wert um 10 erhöht wird, so daß es sich

stets um eine zweistellige Zahl handelt (es gibt 50 mögliche Typnummern/-Namen).

Zeile 13200: Zu der Folge von erstellten Elementen werden nun eine Reihe von Zeichen hinzugefügt, deren Codewert gleich der Position des letzten Zeichens jedes Elementes ist. Am Ende der Zeichenfolge wird ein weiteres Zeichen hinzugefügt, dessen Codewert gleich der Anzahl von Elementen in dem Eintrag ist. Werden die Einträge auf Band gespeichert, so müssen diese Zeichen in Zahlen übersetzt werden, da die Zeichen unter Umständen nicht im Bereich der Zeichen liegen, die in Zeichenform gespeichert werden können.

Testen von Modul 4.2.3

Das Modul kann nicht voll getestet werden, allerdings kann geprüft werden, ob es ausgeführt werden kann, indem temporäre RETURN-Befehle bei 14000 und 15000 eingegeben werden. Danach sollte der Benutzer in der Lage sein, Elemente einzugeben und den Eintrag mit einem Sternchen zu beenden.

MODUL 4.2.4

```
14000 REM#*******************************
14010 REM SUCHROUTINE
14020 REM********************************
14030 IF IT=0 THEN SS=0:RETURN
14040 PO=INT(LOG(IT)/LOG(2)):SS=2↑PO-1
14050 FOR I=PO TO 0 STEP-1
14060 IF A$(SS)<T1$ THEN SS=SS+2↑I
14070 IF A$(SS)>T1$ THEN SS=SS-2↑I
14080 IF SS<0 THEN SS=0
14090 IF SS>IT-1 THEN SS=IT-1
14100 NEXT I:IF A$(SS)<T1$ THEN SS=SS+1
14110 RETURN
```

Ein Standardmodul für die binäre Suche wie in Unifile.

MODUL 4.2.5

```
15000 REM#*******************************
15010 REM EINFUEGEN
15020 REM********************************
15030 IF IT=0 THEN GOTO 15050
15040 FOR I=IT TO SS+1 STEP -1:A$(I)=A$(
```

```
I-1):NEXT
15050 A$(SS)=T1$:IT=IT+1:RETURN
```

Ein direktes Einfüge-Modul.

Testen von Modul 4.2.4 und 4.2.5

Nun sollte der Benutzer in der Lage sein, Elemente einzugeben und sie in das Hauptdatei-Feld (A$) zu speichern. Dies kann nur im Direktmodus geprüft werden.

MODUL 4.2.6

```
19000 REM#******************************
19010 REM NEUE FELDNAMEN EINGEBEN
19020 REM*******************************
19030 FOR I=0 TO 49 STEP 10
19040 PRINT"█████████████████[REVERS EIN]
[ROT]FELDNAMEN"
19050 PRINT "[GRUEN]";:FOR J=1 TO I+10:P
RINT J+1;")";B$(J):NEXT J
19060 PRINT"█[BLAU]MOEGLICHE BEFEHLE:"
19070 PRINT" [SCHWARZ]>[PURPUR]'ZZZ'=ZUM
 MENUE"
19080 PRINT" [SCHWARZ]>[PURPUR]'III'=EIN
GEBEN/LOESCHEN"
19090 PRINT" [SCHWARZ]>[PURPUR]'NNN'=NAE
CHSTE SEITE"
19100 INPUT"[BLAU]BEFEHL:";Q$:
19110 IF Q$="NNN" THEN NEXT I:RETURN
19120 IF Q$<>"III" THEN GOTO 19000
19130 INPUT"[PURPUR]FELD NUMMER:";Q
19140 PRINT"[GRUEN]NAME DES FELDES ODER
[REVERS EIN]RETURN[REVERS AUS] UM ES ZU
 LOESCHEN:";
19150 Q$="":INPUT Q$:B$(Q-1)=Q$:GOTO1904
0
```

Hier handelt es sich um ein neues Modul, mit dem der Benutzer Elementtypen definieren kann. Das Modul zeigt einfach den Inhalt des Feldes B$ in Elfergruppen an und gibt dem Benutzer die Möglichkeit, einen Typnamen in eine bestimmte

Position in das Feld einzugeben. Nachdem er eingegeben wurde, kann ein Typname an ein Element angefügt oder wie unter Modul 4 beschrieben eingegeben werden. Die Typnamen können neu definiert werden, indem einfach ein neuer Name in die Position eingegeben wird, die von einem alten Namen belegt wird. Sie können auch gelöscht werden, indem die RETURN-Taste betätigt wird, wenn nach einem Typnamen gefragt wird.

Testen von Modul 4.2.6

Hier werden einige Typnamen eingegeben und dann zu dem Haupteingabemodul zurückgegangen. Dort wird '↑ ' gefolgt von der Nummer eines definierten Typnamens eingegeben. Die Eingabeaufforderung muß unter dem gewünschten Typnamen wiederholt werden.

MODUL 4.2.7

```
20000 REM#******************************
20010 REM LADEN/SPEICHERN
20020 REM*******************************
20030 PRINT"■[BLAU]KASSETTE EINLEGEN, DA
NN [REVERS EIN]RETURN[REVERS AUS]--"
20040 INPUT"MOTOR HAELT AUTOMATISCH AN:"
;Q$:POKE192,7:POKE1,39
20050 PRINT"■[ROT]MOEGLICHE BEFEHLE:":PR
INT"■[GRUEN] 1)DATEN SPEICHERN"
20055 PRINT"2)DATEN LADEN"
20060 INPUT"■[BLAU]BEFEHL:";Q:ONQGOTO200
70,20150:RETURN
20070 POKE 1,7:FOR I=1 TO 2000:NEXT
20080 OPEN 1,1,1,"UNIFILE":PRINT#1,IT
20090 FOR S1=0 TO IT-1:PRINT#1,FNA(S1)
20100 FOR I=1 TO FNA(S1)-1:PRINT#1,FNB(S
1):NEXT I
20110 PRINT#1,LEFT$(A$(S1),LEN(A$(S1))-F
NA(S1)):NEXT S1
20120 FOR I=0 TO 49:IF B$(I)="" THEN B$(
I)=" "
20130 PRINT#1,B$(I):NEXT
20140 CLOSE1:RETURN
20150 OPEN 1,1,0,"UNIFILE":INPUT#1,IT
20160 FOR S1=0 TO IT-1:INPUT#1,NN
```

```
20170 TT$="":FORI=1TO NN-1:INPUT#1,TT:TT
$=TT$+CHR$(TT):NEXTI:TT$=TT$+CHR$(NN-1)
20180 GET#1,T$:IF T$<>CHR$(13) THEN A$(S
1)=A$(S1)+T$:GOTO 20180
20185 A$(S1)=A$(S1)+TT$:NEXT S1
20190 FOR I=0 TO 49:INPUT#1,B$(I):NEXT
20200 CLOSE1:RETURN
```

Ein Standardmodul einer Datendatei.

MODUL 4.2.8

```
21000 REM#*******************************
21010 REM UNTERPROGRAMME
21020 REM********************************
21030 SS$=MID$(A$(S2),PP+1,FNB(S2)-PP):R
ETURN
21040 FF=0:T1$=RIGHT$(T1$,LEN(T1$)-3):FO
R S2= S1TO IT-1:FOR J=1 TO LEN(A$(I))
21050 IF MID$(A$(S2),J,LEN(T1$))=T1$ THE
N FF=1:S1=S2:J=LEN(A$(S2)):S2=IT-1
21060 NEXT J,S2:RETURN
21070 FF=0:FOR S2=S1 TO IT-1:PP=0:FOR I=
1 TO FNA(S2)-1:GOSUB 21030
21080 IF SS$=T1$ THEN FF=1:S1=S2:I=FNA(S
2)-1:S2=IT-1:GOTO 21120
21090 IF LEN(SS$)<4 THEN 21110
21100 IF MID$(SS$,LEN(SS$)-2,1)="↑" THEN
 SS$=LEFT$(SS$,LEN(SS$)-3):GOTO 21080
21110 PP=FNB(S2)
21120 NEXT I,S2:RETURN
```

Dieses Modul besteht aus drei Routinen, die hier wirtschaftlicher angeordnet sind, da sie von verschiedenen Teilen des Programms aufgerufen werden.

Kommentar

Zeile 21030: Diese Zeile kann aus zwei Schleifen heraus aufgerufen werden: einer S2-Schleife, mit der die Zeile in der Hauptdatei angegeben wird, und einer I-Schleife, mit der die Elementnummer innerhalb des jeweiligen Eintrags angegeben wird. Danach wird Element I in Form von SS$ aus Eintrag S2 ausgegeben.

Zeilen 21040-21060: Entsprechen der besonderen Suchroutine in Unifile.

Zeilen 21070-21120: Eine direkte Suche nach Elementen. Die I-Schleife benutzt FNA, um festzustellen, wie viele Elemente in dem Eintrag vorhanden sind (FNA(S2) der Anzahl von Elementen in Eintrag S2 plus 1 für den Indikator am Ende). Danach wird 21030 aufgerufen, um die einzelnen Elemente auszugeben. Elemente mit einem Typindikator werden mit und ohne Typ-Suffix verglichen.

Testen von Modul 4.2.8

Das Modul kann erst überprüft werden, nachdem das nächste Modul eingegeben wurde.

MODUL 4.2.9

```
17000 REM#********************************
17010 REM SUCHEN
17020 REM********************************
17030 S1=0:FF=0:PRINT"■■■■■■■■■■■■■■■■■■■■■[
ROT][REVERS EIN]SUCHEN"
17040 PRINT"■[BLAU]MOEGLICHE BEFEHLE:"
17050 PRINT"■ [SCHWARZ]>[GRUEN]FUER NORM
ALE SUCHE DAS FELD EINGEBEN"
17060 PRINT" [SCHWARZ]>[GRUEN]DAVOR 'III
' FUER INITIALSUCHE"
17070 PRINT" [SCHWARZ]>[GRUEN]DAVOR 'SSS
' FUER SPEZIELLE SUCHE"
17080 PRINT" [SCHWARZ]>[GRUEN][REVERS EI
N]RETURN[REVERS AUS] FUER DEN ERSTEN DAT
ENSATZ"
17090 PRINT" [SCHWARZ]>[GRUEN]'MMM' FUER
 MEHRFACHE SUCHE"
17100 PRINT "■■[PURPUR]●●●●●●●●●●●●●●●●●
●●●●●●●●●●●●●●●●●●●●●●●"
17110 T1$="":INPUT"[BLAU]SUCHBEFEHL:";T1
$:IFLEFT$(T1$,1)<>"↑"THEN17130
17120 LET TN=VAL(MID$(T1$,2))+10:GOTO 17
110
17130 IF TN<>0 THEN LET T1$=T1$+"↑"+MID$
(STR$(TN),2):TN=0
17140 IF LEFT$(T1$,3)<>"III" THEN 17170
```

```
17150 T1$=RIGHT$(T1$,LEN(T1$)-3):GOSUB 1
4000:S1=SS:IF S1>IT-1 THEN RETURN
17160 T1$="III"+T1$:GOTO 17310
17170 IF LEFT$(T1$,3)<>"SSS" THEN 17200
17180 GOSUB 21040:IF FF=1 THEN T1$="SSS"
+T1$:GOTO 17310
17190 RETURN
17200 IF LEFT$(T1$,3)<>"MMM" THEN 17270
17210 GETZ$:INPUT"[GRUEN]■ANZAHL DER SUC
HDATEN:";NN
17220 FORK=0TONN-1:PRINT"DATUM NUMMER";K
+1;":";:INPUTM$(K):NEXTK
17230 FOR K=0 TO NN-1:T1$=M$(K):S3=S1:GO
SUB21070:IF FF=0 THEN RETURN
17240 IF S3<>S1 THEN 17230
17250 NEXT K
17260 LET T1$="MMM":GOTO 17310
17270 IF T1$="" THEN 17310
17280 GOSUB 21070
17290 IF FF=1 THEN 17310
17300 RETURN
17310 IF S1>IT-1 THEN S1=IT-1
17320 IF IT=0 THEN RETURN
17330 IF S1<0 THEN S1=0
17340 PRINT"■■[ROT]DATENSATZ ";S1+1;":-■
":PP=0:S2=S1:FORI=1TOFNA(S1)-1
17345 IFI/12=INT(I/12)THENINPUT"[SCHWARZ
]WEITER:";TT$
17350 GOSUB 21030:IF LEN(SS$)<4 THEN 173
80
17360 IF MID$(RIGHT$(SS$,3),1,1)<>"↑" TH
EN 17380
17370 PRINT B$(VAL(RIGHT$(SS$,2))-11);":
";:SS$=LEFT$(SS$,LEN(SS$)-3)
17380 PRINT SS$
17390 PP=FNB(S1):NEXT I:S1=S1+1
17400 PRINT"■MOEGLICHE [REVERS EIN][ROT]
SUCH[REVERS AUS][BLAU]BEFEHLE:"
17410 PRINT"■ [SCHWARZ]>MIT [PURPUR][REV
ERS EIN]RETURN[REVERS AUS] ZUM NAECHSTEN
```

```
 DATENSATZ"
17420 PRINT" [SCHWARZ]>[PURPUR]'AAA' UM
ZU AENDERN"
17430 PRINT" [SCHWARZ]>[PURPUR]'CCC' UM
WEITERZUSUCHEN"
17440 PRINT" [SCHWARZ]>[PURPUR]'#' + ZAH
L UM DEN ZAEHLER ZU ERHOEHEN"
17450 PRINT" [SCHWARZ]>[PURPUR]'ZZZ' UM
INS MENUE ZURUECKZUKEHREN"
17460 P$="":INPUT"▒[BLAU]EINGABE:";P$
17470 IF T1$="MMM" AND S1<IT THEN 17230
17480 IF P$="CCC" AND S1<IT THEN 17140
17490 IF P$="" THEN 17310
17500 IF P$="AAA" THEN GOSUB 16000:GOTO
17310
17510 IF P$="ZZZ" THEN RETURN
17520 IF LEFT$(P$,1)="#" THEN S1=S1+VAL(
MID$(P$,2))-1:GOTO 17310
17530 S1=S1-1:GOTO 17310
```

Entspricht dem Suchmodul in Unifile, sieht jedoch die Mehrfachsuche und Typnamen vor.

Kommentar

Zeilen 17110-17130: Hier muß beachtet werden, wie diese Routine feststellt, ob nach einem Element mit einer angefügten Typnummer gesucht wird, und dann die Eingabe unter dieser Typüberschrift anfordert, wobei die Typnummer an das Ende des Elementes angefügt
wird.

Zeilen 17140-17160: Eine erste Suche wie in Unifile.

Zeilen 17170-17190: Besondere Suche, bei der die Routine des vorhergehenden Moduls benutzt wird.

Zeilen 17200-17260: Die neue Mehrfach-Suchroutine. Mit ihr wird der Benutzer aufgefordert, die Anzahl von zu suchenden Elementen anzugeben und danach das einzelne Element einzugeben (Typnummern werden nicht verarbeitet). Eine Suchroutine wird bei 21070-21120 aufgerufen. Bevor jedes Suchelement angegeben wird, wird der Wert des Suchpointers S1 in Form der Variablen S3 aufge-

zeichnet. Kehrt die Routine bei 21070 zu dieser Routine zurück, so wird der Wert von S3 erneut mit S1 verglichen. Unterscheidet sich S1 von S3, so ist klar, daß die beiden Elemente nicht in demselben Eintrag standen. Wird eines der angegebenen Suchelemente das erste Mal gefunden, so wird die Suche auf das erste der angegebenen Suchelemente zurückgesetzt, um zu gewährleisten, daß die ganze Liste mit Suchelementen mit den Elementen in dem Eintrag verglichen wird.

Zeile 17280: Hat die Suche diese Stelle erreicht, so wird davon ausgegangen, daß es sich bei der Eingabe um ein Element handelt, nach dem mit einer normalen Suche gesucht werden muß. In diesem Fall wird die Suchroutine bei 21070 aufgerufen. Hier wird auf die Benutzung des Flag FF in all diesen Suchroutinen hingewiesen. Mit ihm wird angegeben, ob die Suche erfolgreich war.

Zeilen 17340-17390: Am Anfang des ausgewählten Datensatzes beginnend wird die Routine bei 21030 aufgerufen, um einzelne Elemente auszugeben. Typnamen werden ausgedruckt, wenn das '↑ ' Symbol zwei Zeichen vor dem Ende des Elementes steht.

Testen von Modul 4.2.9

Nun sollte der Benutzer in der Lage sein, in den Einträgen vor- und zurückzublättern, und die Suche mit den in dem Kommentar beschriebenen Methoden auszuführen.

MODUL 4.2.10

```
18000 REM#*****************************
18010 REM FILE ZUSAMMENSCHIEBEN
18020 REM******************************
18030 FOR I=S1 TO IT-1:A$(I)=A$(I+1):NEX
T:IT=IT-1:RETURN
```

Mit dieser einzeiligen Routine wird die Datei durchsucht, wenn Löschungen vorgenommen werden.

MODUL 4.2.11

```
16000 REM#*****************************
16010 REM AENDERN EINES DATENSATZES
16020 REM******************************
16030 S1=S1-1:T1$="":NN=-1:TN=0
16040 PP=0:S2=S1:FOR I=1 TO FNA(S1)-1
```

```
16050 PRINT"■■DATENSATZ ";S1+1;":-"
16060 GOSUB 21030:IF LEN(SS$)<4 THEN 160
90
16070 IF MID$(SS$,LEN(SS$)-2,1)<>"↑" THE
N 16090
16080 PRINT B$(VAL(RIGHT$(SS$,2))-11);":
";LEFT$(SS$,LEN(SS$)-3):GOTO 16100
16090 PRINT SS$
16100 PRINT"[BLAU]■■■■■■MOEGLICHE BEFEHL
E:"
16110 PRINT" [SCHWARZ]>[GRUEN][REVERS EI
N]RETURN[REVERS AUS] LAESST DAS FELD UNG
EAENDERT"
16120 PRINT" [SCHWARZ]>[GRUEN]NEUE FELDE
R MIT '*' BEENDEN"
16130 PRINT" [SCHWARZ]>[GRUEN]DAS GEAEND
ERTE FELD WIRD ANGEZEIGT";
16140 PRINT" [SCHWARZ]>[GRUEN]'ZZZ' KEHR
T OHNE AENDERUNGEN ZURUECK"
16150 PRINT" [SCHWARZ]>[GRUEN]'DDD' LOES
CHT DEN GANZEN DATENSATZ"
16160 PRINT" [SCHWARZ]>[GRUEN]'RRR' LOES
CHT DIESES FELD AUS DEM":PRINT"        D
ATENSATZ"
16170 Q$="":INPUT"[ROT]EINGABE:";Q$
16180 IF Q$="ZZZ" THEN RETURN
16190 IF Q$="RRR" THEN GOTO 16300
16200 IF Q$="DDD" THEN GOSUB 18000:RETUR
N
16210 IF LEN(T1$)+LEN(Q$)+NN+2<255 THEN
GOTO 16230
16220 PRINT"[ROT]DATENSATZ ZU LANG":FORJ
=1TO3000:NEXT:RETURN
16230 EX=0:IF RIGHT$(Q$,1)="*" THEN EX=1
:Q$=LEFT$(Q$,LEN(Q$)-1)
16240 IF Q$="" THEN T1$=T1$+SS$:TI%(NN+1
)=LEN(T1$):NN=NN+1:GOTO 16300
16250 IF LEFT$(Q$,1)<>"↑" THEN 16280
16260 TN=VAL(MID$(Q$,2))+10:PRINT B$(TN-
11);":";:Q$="":INPUT Q$
```

```
16270 Q$=Q$+"↑"+MID$(STR$(TN),2)
16280 T1$=T1$+Q$:TI%(NN+1)=LEN(T1$):NN=N
N+1
16290 IFEX<>1THEN16050
16300 PP=FNB(S1):NEXT I
16310 FOR I=0 TO NN:T1$=T1$+CHR$(TI%(I))
:NEXT:T1$=T1$+CHR$(NN+1)
16320 PRINT"▩[SCHWARZ]BITTE WARTEN":GOSU
B18000:GOSUB14000:GOSUB15000:RETURN
```

Entspricht dem Unifile-Änderungsmodul.

Kommentar

Zeile 16230: Wird ein Element eingegeben, das mit einem * beendet wird, so bedeutet dies, daß ein neues Element vor dem gerade angezeigten Element in den Eintrag gesetzt werden muß. Die Variable EX (EXtra) wird auf Eins gesetzt, um anzugeben, daß dies geschehen ist. In Zeile 16290 wird mit dieser Variablen gewährleistet, daß das angezeigte Element nicht verloren ist.

Testen von Modul 4.2.11

Nun sollte der Benutzer in der Lage sein, Einträge zu löschen, Elemente zu ändern oder einzufügen. Wird dieses Modul einwandfrei ausgeführt, so ist das Programm zur Benutzung bereit.

Zusammenfassung

Wird berücksichtigt, daß die Anwendungen unterschiedlicher Natur sind, so hat dieses Programm sämtliche Vorteile des ursprünglichen Unifile-Programms, und ich hoffe, daß Sie es ebenfalls nützlich finden.
Darüber hinaus hoffe ich, daß die Eingabe des Programms Ihnen Einblick in die Vorteile der modularen Programmierung gewährt hat. Aufbauend auf den ursprünglichen Unifile- Moduln konnte die Originalversion dieses Programms in weniger als einem Morgen geschrieben werden, da aufgrund der eindeutigen Struktur eines modularen Programms absolut deutlich wird, an welcher Stelle Änderungen vorgenommen werden müssen.
Es wird Zeit und Mühe bei der Programmierung gespart, wenn die Programme in eindeutig definierten VV Funktionseinheiten erstellt werden, es sei denn, es steht sehr wenig Speicherplatz zur Verfügung. Dadurch werden die Programme nicht nur einfacher lesbar, sondern wird auch die Wahrscheinlichkeit erhöht, daß dieselbe Routine von verschiedenen Teilen des Programms aufgerufen werden

kann. Außerdem wird der Austausch der Funktionen vereinfacht, die Sie zu einem späteren Zeitpunkt verbessern können. Und schließlich können ganze Abschnitte problemlos zur späteren Benutzung in anderen Anwendungen aus dem Programm herausgenommen werden.

Ein Schritt weiter

1. Die Mehrfach-Suchroutine sieht die Angabe von Typnamen der Elemente nicht vor. Die Zeilen 17110-17130 sind ein klares Beispiel dafür, wie eine derartige Möglichkeit geschaffen werden könnte.
2. In professionellen Datenbanken besteht im allgemeinen immer die Möglichkeit, nach Einträgen zu suchen, bei denen beispielsweise vier von insgesamt acht Suchkriterien vorhanden sind. Mit einigen kleinen Änderungen könnte das vorliegende Programm dieses Ziel ebenfalls erreichen.

4.3 NNUMBER

Nachdem nun zwei Programme eingegeben wurden, die eine Vielzahl von Anforderungen im Bereich der Speicherung nicht numerischer Daten abdecken, wollen wir uns jetzt den Problemen der Verfolgung von Zahlen zuwenden. Obwohl die meisten numerischen Anwendungen für ein bestimmtes Problem geschrieben werden müssen, ist NNumber (Abkürzung für Name und Number) den beiden Unifile-Programmen insofern sehr ähnlich, als es ein Universalprogramm für Anwendungen ist, bei denen die Namen von Elementen und die mit ihnen verknüpften Mengeneinheiten gespeichert und die Elemente in verschiedenen Mengen addiert werden müssen. Sollten Sie der Auffassung sein, daß Sie eine derartige Anwendung nie benötigen, so sollte ich vielleicht sagen, daß die Idee für das Programm aus einem Kalorienzähler entstand, mit dem der Benutzer ein Verzeichnis von bis zu 500 Speisen speichern und den Kalorienwert der Nahrungsmittelaufnahme eines Tages oder einer Woche berechnen kann. Das vorliegende Programm kann ohne Änderung ebenso problemlos Rechnungen erstellen wie Kalorien zählen.
Da der Stil des Programms den beiden Unifile-Programmen sehr ähnlich ist, und da viele der Funktionen gleich sind, wurden die Kommentare und die Testempfehlungen weitestgehend abgekürzt.

NNumber: Variablentabelle

A$(488,1)	Hauptverzeichnis-Feld
C(499)	Werte, die mit den im Hauptverzeichnis angegebenen Einheiten verknüpft sind

CT	Temporäre Variable, mit der die Summe der Elemente in der aktuellen Liste kumuliert wird
CU	Anzahl von Elementen in der aktuellen Liste
NN$	Allgemeiner Name für die aufgezeichneten Elemente
IN	Initialisierungs-Flag
IT	Anzahl von Elementen, die in dem Hauptverzeichnis gespeichert sind
NN	Temporärer Speicher für den Wert, der mit dem neuen eingegebenen Element verknüpft ist
PO	Mit dieser Variablen wird die Anzahl von Vergleichen in dem binären Suchmodul festgelegt
QQ$	Allgemeiner Name für Mengen, die mit den aufgezeichneten Elementen verknüpft sind
R$	Separator, der bei der Sicherung der Datendatei benutzt wird
SS	Such-Pointer für die binäre Suche
T(49)	Werte, die mit den in T$ gespeicherten Elementen verknüpft sind
T$(49,1)	Speicherung der aktuellen Liste
T1$	Temporäre Speicherung des eingegebenen Elementnamens
T2$	Temporäre Speicherung von Einheiten, die mit T1$ verknüpft sind

MODUL 4.3.1

```
11000 REM#******************************
11010 REM MENUE
11020 REM*******************************
11030 POKE 53281,13:PRINT "
 [ROT][REVERS EIN]NAME AND NUMBER[REVERS
 AUS]"
11040 PRINT "[BLAU]MOEGLICHE BEFEHLE:"

11050 PRINT "[GRUEN]    1)LISTE AUSGEBE
N"
11060 PRINT "    2)EINGABEN IN DIE LISTE"

11070 PRINT "    3)NEUE LISTE BEGINNEN"
11080 PRINT "    4)AUS DER LISTE LOESCHEN
"
11090 PRINT "    5)DATEI ERWEITERN"
11100 PRINT "    6)DATEI AUSGEBEN"
11110 PRINT "    7)LADEN/SPEICHERN"
11120 PRINT "    8)INITIALISIEREN"
11130 PRINT "    9)BEENDEN"
```

```
11140 INPUT "[BLAU]BEFEHL:";Z:PRINT ""
;
11150 IF IN<>0 OR Z=8 OR Z=9 THEN GOTO 1
1180
11160 PRINT "[ROT]NOC
H NICHT INITIALISIERT"
11170 FOR I=1 TO 2000:NEXT:GOTO 11000
11180 IF IT>0 OR Z=2 OR Z=3 OR Z=5 OR Z=
7 OR Z=8 OR Z=9 THEN 11210
11190 PRINT "[ROT]NOCH
 KEINE DATEN VORHANDEN"
11200 FOR I=1 TO 2000:NEXT:GOTO 11000
11210 ON Z GOSUB 13000,14000,14120,19000
,15000,18000,20000,12000,11230
11220 GOTO 11000
11230 PRINT "[RO
T]NAME AND NUMBER"
11240 PRINT "[GRUEN]PROGRA
MM BEENDET":END
```

Standard-Menümodul.

MODUL 4.3.2

```
12000 REM#******************************
12010 REM INITIALISIEREN
12020 REM*******************************
12030 CLR:DIM A$(499,1),C(499),T$(49,1),
T(49):CURR=0:IT=0:IN=1:R$=CHR$(13)
12035 INPUT "[ROT]MOECHTEN SIE VON KASS
ETTE LADEN (J/N):";Q$:IF Q$="J" THEN 110
00
12040 INPUT "[BLAU]NAME DER OBJEKTE:";N
N$
12050 INPUT "MENGENEINHEIT:";QQ$:GOTO 1
1000
```

Initialisierung von Variablen. Mit diesem Modul kann der Benutzer auch den Namen des zu speichernden Elementtyps und den allgemeinen Namen der Einheiten angeben, z. B. Nahrungsmittel/Einheiten, Produkt/Verpackung.

MODUL 4.3.3.

```
15000 REM#******************************
15010 REM DATEI ERWEITERN
15020 REM*******************************
15030 IF ITEMS<500 THEN 15050
15040 PRINT "█[ROT]██KEIN PLATZ MEHR IN
DER DATEI":FOR I=1 TO 2000:NEXT:RETURN
15050 PRINT "█████████[REVERS EIN][BRAUN]
NEUE OBJEKTE FUER DIE DATEI[REVERS AUS]"

15060 PRINT "██[ROT]";NN$;:INPUT ":(NAME
 ODER 'ZZZ' FUER MENUE):";T1$
15070 IF T1$="ZZZ" THEN RETURN
15080 PRINT "█";QQ$;:INPUT ":";T2$
15090 PRINT "█MENGE PRO ";T2$;:INPUT NN
15100 INPUT "█[BLAU]EINGABE KORREKT (J/N
):";Q$:IF Q$="N" THEN GOTO 15050
15110 GOSUB 16000:GOSUB 17000:IT=IT+1:GO
TO 15050
```

Das Eingabemodul für das Hauptverzeichnis. Nachdem der allgemeine Name für die Elemente und deren Einheiten eingegeben wurde, wird der Benutzer aufgefordert, den Elementnamen, den Einheitennamen und die Grundmenge pro Einheit (d.h. Kalorien, Preis, Menge) einzugeben.

MODUL 4.3.4

```
16000 REM#******************************
16010 REM SUCHROUTINE
16020 REM*******************************
16030 IF IT=0 THEN SS=0:RETURN
16040 PO=INT(LOG(IT)/LOG(2)):SS=2↑PO-1
16050 FOR I=PO TO 0 STEP -1
16060 IF A$(SS,0)<T1$ THEN SS=SS+2↑I
16070 IF A$(SS,0)>T1$ THEN SS=SS-2↑I
16080 IF SS<0 THEN SS=0
16090 IF SS>IT-1 THEN SS=IT-1
16100 NEXT I
16110 IF A$(SS,0)<T1$ THEN SS=SS+1
16120 RETURN
```

Ein Standardmodul für die binäre Suche.

MODUL 4.3.5

```
17000 REM#*****************************
17010 REM OBJEKT EINFUEGEN
17020 REM******************************
17030 IF IT=0 THEN GOTO 17060
17040 FOR I=IT TO SS+1 STEP-1:A$(I,0)=A$
(I-1,0):A$(I,1)=A$(I-1,1)
17050 C(I)=C(I-1):NEXT
17060 A$(SS,0)=T1$:A$(SS,1)=T2$:C(SS)=NN
:RETURN
```

Ein Standard-Einfügemodul.

MODUL 4.3.6

```
18000 REM#*****************************
18010 REM DATEI AUSGEBEN
18020 REM******************************
18030 SS=0:T1$="*1"
18040 PRINT "■■■■■■■■■■■■■■■■[REVERS EIN
][ROT]SUCHEN[REVERS AUS]"
18050 PRINT "[BLAU]■OBJEKTNUMMER:";SS+1
18060 PRINT "■[ROT]";NN$;":";A$(SS,0)
18070 PRINT "■";QQ$;":";A$(SS,1)
18080 PRINT "■MENGE PRO ";A$(SS,1);":";C
(SS)
18090 PRINT "■[GRUEN]●●●●●●●●●●●●●●●●●●●●
●●●●●●●●●●●●●●●●●●●●●●●●●"
18100 PRINT "[SCHWARZ]MOEGLICHE BEFEHLE:
"
18110 PRINT "■[GRUEN]    >OBJEKT NACH DEM
 GESUCHT WERDEN SOLL"
18120 PRINT "    >'*'+ NUMMER UM DEN ZAEH
LER ZU ERHOE-          HEN
18130 PRINT "    >'DDD' UM DAS OBJEKT ZU
LOESCHEN"
18140 PRINT "    >'ZZZ' UM INS MENUE ZURU
ECKZUKEHREN"
```

```
18150 T1$="":INPUT "■[PURPUR]BEFEHLE:";T
1$
18160 IF T1$<>"DDD" THEN 18190
18170 FOR I=SS TO IT-1:A$(I,0)=A$(I+1,0)
:A$(I,1)=A$(I+1,1):C(I)=C(I+1):NEXT
18180 IT=IT-1:GOTO 18040
18190 IF T1$="ZZZ" THEN RETURN
18200 IF LEFT$(T1$,1)<>"*" THEN GOSUB 16
000:GOTO 18220
18210 SS=SS+VAL(MID$(T1$,2))
18220 IF SS>IT-1 THEN SS=IT-1
18230 IF SS<0 THEN SS=0
18240 GOTO 18040
```

Das Haupt-Suchmodul des Benutzers.

Testen der Moduln 4.3.1–4.3.6

Der Benutzer sollte nun in der Lage sein, Daten einzugeben und mit Hilfe des Suchmoduls zu prüfen, ob diese richtig (nach Elementnamen sortiert) in das Hauptverzeichnis eingegeben wurden.

MODUL 4.3.7

```
20000 REM#******************************
20010 REM LADEN/SPEICHERN
20020 REM*******************************
20030 PRINT "■[ROT]KASSETTE EINLEGEN, DA
NN [REVERS EIN]RETURN[REVERS AUS]--"
20040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
20050 PRINT "■[SCHWARZ]MOEGLICHE BEFEHLE
:":PRINT "[purpur]1)daten speichern"
20055 PRINT "■2)DATEN LADEN"
20060 INPUT "■[SCHWARZ]BEFEHLE:";Q: ON Q
 GOTO 20070,20140:RETURN
20070 POKE 1,7:FOR I=1 TO 2000:NEXT
20080 OPEN 1,1,2,"NNUMBER":PRINT#1,NN$,R
$,QQ$,R$,CU,R$,IT
20090 IF CU=0 THEN 20110
20100 FOR I=0 TO CU-1:PRINT#1,T$(I,0),R$
```

```
,T$(I,1),R$,T(I):NEXT
20110 IF IT=0 THEN 20130
20120 FOR I=0 TO IT-1:PRINT#1,A$(I,0),R$
,A$(I,1),R$,C(I):NEXT
20130 CLOSE1:RETURN
20140 OPEN 1,1,0,"NNUMBER":INPUT#1,NN$,Q
Q$,CU,IT
20150 IF CU=0 THEN 20170
20160 FOR I=0 TO CU-1:INPUT#1,T$(I,0),T$
(I,1),T(I):NEXT
20170 IF IT=0 THEN 20190
20180 FOR I=0 TO IT-1:INPUT#1,A$(I,0),A$
(I,1),C(I):NEXT
20190 CLOSE1:RETURN
```

Ein Standardmodul der Datendatei. Hier werden die Daten gespeichert.

MODUL 4.3.8

```
14000 REM#*****************************
14010 REM LISTE ERWEITERN
14020 REM******************************
14030 IF CU=50 THEN PRINT "[ROT]
         LISTE IST VOLL.":RETURN
14040 PRINT "[ROT][REVERS EIN]ER
WEITERN DER LISTE[REVERS AUS]"
14050 PRINT "[GRUEN]";NN$;" ('ZZZ' FUE
R MENUE):";:INPUT T1$:IF T1$="ZZZ" THEN
RETURN
14060 GOSUB 16000:IF A$(SS,0)=T1$ THEN G
OTO 14080
14070 PRINT"[BLAU]UNBEKANNT, BITTE UEB
ERPRUEFEN.":FORI=1TO2000:NEXT:RETURN
14080 PRINT "";QQ$;":";A$(SS,1):INPUT "
menge:";q
14090 INPUT "[PURPUR]EINGABE KORREKT (
J/N):";Q$:IF Q$="N" THEN 14000
14100 T$(CU,0)=A$(SS,0):T$(CU,1)=STR$(Q)
+"  "+A$(SS,1):T(CU)=Q*C(SS):CU=CU+1
14110 GOTO 14000
```

Bis jetzt konnte der Benutzer Angaben in das Hauptverzeichnis eingeben und die Daten verarbeiten. Der wichtigste Punkt des Programms besteht jedoch darin, was aus dem Verzeichnis ausgegeben und in einer temporären Liste gespeichert werden kann, die als aktuelle Liste bezeichnet wird. Am Anfang dieses Moduls kann der Benutzer ein Element in dem Verzeichnis benennen und angeben, wie viele Einheiten dieses Elementes zu der aktuellen Liste hinzugefügt werden sollen. Das binäre Suchmodul wird aufgerufen, um zu prüfen, ob die Elementeingabe tatsächlich in dem Verzeichnis steht.

MODUL 4.3.9

```
13000 REM#*****************************
13010 REM LISTE AUSGEBEN
13020 REM******************************
13030 IF CU=0 THEN RETURN
13040 PRINT"■":CT=0:FOR I=0 TO CU-1:PRIN
T "■[BLAU]";NN$;": ";T$(I,0)
13050 PRINT "■";QQ$;": ";T$(I,1):PRINT "■
MENGE:";T(I)
13060 PRINT "■[GRUEN]oooooooooooooooooo
oooooooooooooooooooooo"
13070 INPUT "[REVERS EIN]RETURN[REVERS A
US] FUER DAS NAECHSTE OBJEKT:";Q$:PRINT
"■■"
13080 PRINT "
        "
13090 CT=CT+T(I):NEXT I
13100 PRINT "[PURPUR]■TOTAL:[SCHWARZ]";C
T
13110 INPUT "■[GRUEN][REVERS EIN]RETURN[
REVERS AUS] UM INS MENUE ZURUECKZUKEHREN
:";Q$:RETURN
```

Nachdem Elemente in die aktuelle Liste eingegeben wurden, kann die aktuelle Liste mit diesem Modul auf dem Bildschirm ausgegeben werden. Mit ihm kann eine Gesamtsumme der Werte erstellt werden, die mit den Elementen in der aktuellen Liste verknüpft sind.

MODUL 4.3.10

```
19000 REM#*****************************
19010 REM AUS DER LISTE LOESCHEN
```

```
19020 REM*******************************
19030 IF CU=0 THEN RETURN
19040 FOR I=0 TO CU-1:Q$="":PRINT "[BLAU
]";t$(i,0):print "";t$(i,1)
19050 INPUT"[GRUEN]■DDD=LOESCHEN[REVERS
EIN] [REVERS AUS]RETURN=NAECHSTES[REVERS
 EIN] [REVERS AUS]ZZZ=MENUE:";Q$
19055 IF Q$="ZZZ" THEN RETURN
19060 IF Q$<>"DDD" THEN NEXT I:RETURN
19070 FOR J=I TO CU-1:T$(J,0)=T$(J+1,0):
T$(J,1)=T$(J+1,1):T(J)=T(J+1):NEXT
19080 CU=CU-1:RETURN
19090 STOP
```

Dieses Modul ermöglicht dem Benutzer, durch die Einträge in der aktuellen Liste zu blättern und sie nach Belieben zu löschen. Es gibt keine komplexen Suchfunktionen, sondern hier wird einfach Element für Element durchgeblättert.

Testen der Moduln 4.3.8–4.3.10

Nun sollte der Benutzer in der Lage sein, eine aktuelle Liste zu erstellen, diese anzuzeigen und nach Belieben Elemente aus der Liste zu löschen.

MODUL 4.3.11

```
14120 REM #******************************
14130 REM NEUE LISTE BEGINNEN
14140 REM *******************************
14150 FOR I=1 TO 50:T$(I,0)="":T$(I,1)="
":T(I)=0:CU=0:RETURN
```

Da die aktuelle Liste nur als temporäre Liste gedacht ist, die häufig zurückgesetzt werden kann, löscht dieses Modul die Felder der aktuellen Liste und setzt den aktuellen List-Pointer auf Null. Wird das Modul richtig ausgeführt, so ist das Programm zur Benutzung bereit.

Zusammenfassung

Dieses Programm ist ein weiteres Beispiel für die Leistungsfähigkeit der modularen Programmierung. Auch wenn die Anwendung sehr verschieden ist, wurden viele der Moduln mit oder ohne Änderung aus den beiden vorherigen Programmen genommen.

Mit fortlaufender Erfahrung als Programmierer wird der Benutzer schnell feststellen, daß eine Sammlung von Methoden zur Ausführung von Aufgaben, die in sauber voneinander getrennte funktionale Einheiten geschrieben werden, sogar noch wichtiger ist als eine Sammlung von Programmen. Eine Bibliothek mit Programmen wird immer gute Dienste leisten, bis eine neue Anwendung benötigt wird. Eine aus einwandfrei arbeitenden Moduln zusammengestellte Bibliothek verhilft dem Benutzer zur problemlosen Lösung dieser neuen Anwendungen. Neue Methoden finden Sie in Zeitschriften und Büchern wie diesem (auch wenn dieses natürlich einmalig ist). Deshalb wird empfohlen, die neuen Methoden zu verfolgen, wenn sie gut erscheinen, auch wenn ihre Relevanz im Augenblick nicht sichtbar wird. Nach einer oder zwei Wochen wird man oft feststellen, daß sie genau das sind, wonach gesucht wird, um das Programm zu vervollständigen, das im Augenblick so viele Probleme schafft.

KAPITEL 5

SCHULE ZU HAUSE

Ein Bereich, in dem sich die Mikrocomputer erst zu bewähren beginnen, ist der Bereich der Bildung und Ausbildung. Keine Schule ist heute ohne einen oder zwei Computer als komplett zu bezeichnen. Das computerunterstützte Lernen ist jedoch nicht nur in der Schule von Bedeutung- Computerleistung zu einem vernünftigen Preis bedeutet, daß all seine Vorteile auch heute zu Hause ausgenützt werden können. In diesem Kapitel werden drei Programme vorgestellt, die ein Beispiel für Anwendungen im Erziehungsbereich darstellen, unabhängig vom Alter des Benutzers.

5.1 MULTIQ

Dieses Programm ist eigentlich mein liebstes. Als ich es geschrieben habe, war ich mit meiner Arbeit überaus zufrieden, da sie die gestellte Aufgabe voll erfüllt. Erst als ich eine große Anzahl von Fragen eingegeben und diese mit verschiedenen Personen ausprobiert habe, habe ich festgestellt, daß das Lernen mit derartigen Programmen ebenso zur Sucht werden kann, wie andere Computerspiele.
Wie Unifile ist dieses Programm ein Chamäleon, das seine Farbe je nach Bedarf ändert. Einmal kann es als Lehrer für Französisch benutzt werden, und wenige Minuten später beantwortet es komplexe Fragen über die Geschichte des 19. Jahrhunderts. Dieses Programm soll all dies und noch mehr ermöglichen, ohne daß Änderungen an dem Programm selbst erforderlich sind.

Multiq: Variablentabelle

AA	Temporäre Variable, in der die vom Benutzer ausgewählte Antwort gespeichert wird
A$(1,499)	Hauptfeld mit Fragen und Antworten
D(1,9)	Zeiger zum Anfang von Gruppen mit Elementtypen in des Hauptfeldes
D$(9)	Feld mit Elementtypen
IT	Anzahl von Elementen, die in das Hauptfeld gespeichert sind
NA$(1)	Feld mit allgemeinen Namen für Fragen und Antworten
P1,P2	Zeiger zu dem Dateibereich, auf den zur Generierung von Fragen Bezug genommen wird
	Zeiger zu der falschen Antwort, die aus der Matrix ausgewählt wurde
PP	Wird beim Erstellen eines Tests mit mehreren Auswahlmöglichkeiten be-
Q(4)	nutzt

Q1	Position der zufälligen Antwort, die aus der Datei geholt wurde
Q2	Position der richtigen Antwort in dem möglichen Antwortfeld Q
QT	Gesamte gestellte Fragen
QU	Indikator zu dem Fragentyp, der in dem Modul zur Generierung von Fragen benötigt wird
R$	Separator in der Datendatei
RR	Richtige Antworten
SU	Temporäre Variable, die bei der Berechnung von Gruppen in dem Feld D benutzt wird
TY	Anzahl von eingegebenen Typnamen

MODUL 5.1.1

```
11000 REM#*****************************
11010 REM MENUE
11020 REM******************************
11030 POKE 53281,15:PRINT "[BLAU]
      [REVERS EIN]MULTIQ"
11040 PRINT "MOEGLICHE BEFEHLE:"
11050 PRINT " [GRAU 2]1)NEUE DATEN EING
EBEN"
11060 PRINT " 2)SUCHEN/LOESCHEN"
11070 PRINT " 3)NEUE FRAGENBEREICHE DEFI
NIEREN"
11080 PRINT " 4)FRAGEN GENERIEREN"
11090 PRINT " 5)ANZEIGEN ODER LOESCHEN D
ER PUNKTZAHL"
11100 PRINT " 6)LADEN/SPEICHERN"
11110 PRINT " 7)INITIALISIEREN"
11120 PRINT " 8)BEENDEN"
11130 INPUT "[HELLBLAU]IHRE EINGABE:";Z
:PRINT "";
11140 ON Z GOSUB 13000,16000,12100,17000
,18000,19000,12000,11150:GOTO 11000
11150 PRINT "[BLAU]
     [REVERS EIN]PROGRAMM BEENDET":END
```

Ein Standard-Menümodul.

MODUL 5.1.2

```
12000 REM#*****************************
12010 REM INITIALISIEREN
12020 REM******************************
12030 CLR:DIM NA$(1),Q(4),A$(1,500),D$(1
0),D(1,10):R$=CHR$(13)
12040 INPUT "■[SCHWARZ]LADEN SIE VON KAS
SETTE (J/N):";Q$:IF Q$="J" THEN 11000
12050 PRINT "■■[BLAU]■■■■■■■■■■■[REVERS
EIN]STRUKTUR DES TESTS"
12060 PRINT "■[ORANGE]ALLGEMEINER TITEL
FUER DIE ANTWORTEN:":INPUT NA$(0)
12070 PRINT "■[BRAUN]ALLGEMEINER TITEL F
UER DIE FRAGEN:":INPUT NA$(1)
12080 INPUT "■[HELLROT]EINGABE KORREKT (
J/N):";Q$:IF Q$="N" THEN 12050
12090 D$(0)="KEINE ART SPEZIFIZIERT":TY=
0
12100 PRINT "■■[BLAU]■■■■■■■■■■■■■FRAGEN
BEREICHE"
12110 PRINT "■[ROT]GEBEN SIE 'ZZZ' EIN U
M INS MENUE ZU-    RUECKZUKEHREN."
12120 PRINT "■[PURPUR]BIS JETZT EINGEBEN
E BEREICHE:-";:IF TY=0 THEN PRINT "[SCHW
ARZ]KEINE"
12130 IF TY<>0 THEN PRINT "■":FOR I=1 TO
 TY:PRINT "[ORANGE]";I;"- [BRAUN]";D$(I)
:NEXT
12140 INPUT "■[BLAU]NEUER FRAGENBEICH:";
Q$:IF Q$ ="ZZZ" THEN 11000
12150 INPUT "■[SCHWARZ]KORREKT (J/N):";Q
1$:IF Q1$ ="N" THEN 12100
12160 IF TY = 10 THEN PRINT "■[ROT]KEIN
PLATZ MEHR ! MAX. 10 BEREICHE !"
12165 IF TY = 10 THEN FOR I=1 TO 2000:NE
XT:GOTO 11000
12170 TY=TY+1:D$(TY)=Q$:GOTO12100
```

Dieses Modul initialisiert die Hauptvariablen und ermöglicht dem Benutzer die Angabe des allgemeinen Namens für Fragen und Antworten. Der Benutzer kann

auch bis zu neun Typnamen angeben, mit denen die Tests später schwieriger gestaltet werden.

MODUL 5.1.3

```
13000 REM#*****************************
13010 REM EINGABE NEUER DATEN
13020 REM******************************
13030 PRINT "▚▣[BLAU]███████████████[REVE
RS EIN]NEUE DATEN":PRINT "[orange]'zzz'
=ZURUECK ZUM MENUE"
13040 IF IT>=500 THEN PRINT "▣[ROT]KEIN
PLATZ FUER WEITERE DATEN !"
13045 IF IT>=500 THEN FOR I=1 TO 2000:NE
XT:RETURN
13050 PRINT "▣[SCHWARZ]";NA$(0);":";:INP
UT T1$:IF T1$="ZZZ" THEN 13160
13060 PRINT "▣[HELLROT]";NA$(1);":";:INP
UT T2$:IF T2$="ZZZ" THEN RETURN
13070 IF TY=0 THEN T=0:GOTO 13100
13080 PRINT "[BRAUN]BEREICH:":FOR I=1 TO
 TY:PRINT I;D$(I):NEXT I
13090 INPUT "▣[BLAU]WELCHES:";T
13100 PRINT "▣[BLAU]███████████████[REVERS
 EIN]NEUE DATEN"
13110 PRINT "▣[ROT][REVERS EIN]";NA$(0);
"[REVERS AUS]:";T1$:PRINT "▣[BRAUN][REVE
RS EIN]";NA$(1);"[REVERS AUS]:";T2$
13120 PRINT "▣[GRUEN][REVERS EIN]BEREICH
[REVERS AUS]";D$(T):INPUT "[schwarz]ein
GABEN RICHTIG (J/N):";Q$
13130 IF Q$<>"J" THEN 13000
13140 D(0,T)=D(0,T)+1:T1$=CHR$(48+T)+T1$
13150 GOSUB 14000:GOSUB 15020:GOTO 13000
13160 SU=0:FOR I=0 TO 9:D(1,I)=SU:SU=SU+
D(0,I):NEXT:RETURN
```

Dieses Modul akzeptiert die Eingabe von Fragen und Antworten unter den vom Benutzer angegebenen Überschriften, wobei ein Typ angefügt werden kann, wenn Typen eingegeben wurden.

Kommentar

Zeile 13070: Ist TY gleich 1, so wurden keine Typnamen eingegeben und der Typ wird auf 'Untyped' gesetzt.

Zeile 13140: Das entsprechende Element in der ersten Spalte des Feldes D wird um 1 erhöht, wobei die Tatsache berücksichtigt wird, daß die Typgruppe erhöht wurde. Die Typnummer wird in Form eines Zeichens zwischen 0 und 9 vor die Antwort gesetzt.

Zeile 13160: Die zweite Spalte des Feldes D wird justiert, so daß sie die Anfangsposition jeder Typgruppe enthält.

MODUL 5.1.4

```
14000 REM#*****************************
14010 REM SUCHROUTINE
14020 REM******************************
14030 IF IT=0 THEN SS=0:RETURN
14040 PO=INT(LOG(IT)/LOG(2)):SS=2↑PO-1
14050 FOR I=PO TO 0 STEP-1
14060 IF A$(0,SS)<T1$ THEN SS=SS+2↑I
14070 IF A$(0,SS)>T1$ THEN SS=SS-2↑I
14080 IF SS<0 THEN SS=0
14090 IF SS>IT-1 THEN SS=IT-1
14100 NEXT I:IF A$(0,SS)<T1$ THEN SS=SS+
1
14110 RETURN
```

Ein Standardmodul für die binäre Suche. Da die Elemente nach Antwort sortiert werden und bei den Antworten das Zeichen für die Typengruppe vorangestellt ist, wird in Wirklichkeit zuerst nach Typ sortiert - Gruppen ohne Typangabe stehen als erstes in der Datei.

MODUL 5.1.5

```
15000 REM#******************************
15010 REM EINFUEGEN
15020 REM*******************************
15030 IF IT=0 THEN 15060
15040 FOR I=IT TO SS+1 STEP -1
15050 FOR J=0 TO 1:A$(J,I)=A$(J,I-1):NEX
```

```
T J,I
15060 A$(0,SS)=T1$:A$(1,SS)=T2$:IT=IT+1:
RETURN
```

Ein Standard-Einfügemodul.

Testen der Moduln 5.1.1–5.1.5

Nun sollte der Benutzer in der Lage sein, Daten einzugeben und sie nach Typnummer sortiert in den Hauptfeldern zu speichern.

MODUL 5.1.6

```
19000 REM#******************************
19010 REM LADEN/SPEICHERN
19020 REM*******************************
19030 PRINT "▣[BLAU]KASSETTE EINLEGEN, D
ANN [REVERS EIN]RETURN[REVERS AUS]--"
19040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
19050 PRINT "▣[ROT]MOEGLICHE BEFEHLE:":P
RINT "[GRUEN] 1)DATEN SPEICHERN",,," 2)D
ATEN LADEN"
19060 INPUT "[SCHWARZ]BEFEHL:";Q:ON Q GO
TO 19070,19130:RETURN
19070 POKE 1,7:FOR I=1 TO 2000:NEXT
19080 OPEN1,1,2,"MULTIQ":PRINT#1,IT;R$;T
Y
19090 FOR I=0 TO TY:PRINT#1,D$(I);R$;D(0
,I);R$;D(1,I):NEXT
19100 FOR I=0 TO IT:PRINT#1,A$(0,I);R$;A
$(1,I):NEXT
19110 PRINT#1,NA$(0);R$;NA$(1)
19120 CLOSE 1:RETURN
19130 OPEN 1,1,0,"MULTIQ":INPUT#1,IT,TY
19140 FOR I=0 TO TY:INPUT#1,D$(I),D(0,I)
,D(1,I):NEXT
19150 FOR I=0 TO IT:INPUT#1,A$(0,I),A$(1
,I):NEXT
19160 INPUT#1,NA$(0),NA$(1)
19170 CLOSE 1:RETURN
```

Ein Standardmodul der Datendatei.

MODUL 5.1.7

```
16000 REM#******************************
16010 REM SUCHEN/LOESCHEN
16020 REM*******************************
16030 SS=0
16040 PRINT "[ROT][RE
VERS EIN]SUCHEN"
16050 IF SS>IT-1 THEN SS=IT-1
16060 IF SS<0 THEN SS=0
16070 PRINT "[GRUEN]AN
ZAHL FRAGEN=";IT
16080 PRINT "[BLAU]MOEGLICHE BEFEHLE:"
16090 PRINT "[SCHWARZ] >[PURPUR][REVERS
 EIN]RETURN[REVERS AUS]=WEITER ZUR NAECH
STEN FRAGE"
16100 PRINT "[SCHWARZ] >[PURPUR]POS/NEG
NUMMER UM DEN ZAEHLER ZU VER-   AENDERN"
16110 PRINT "[SCHWARZ] >[PURPUR]'DDD'=LO
ESCHEN DIESER FRAGE"
16120 PRINT "[SCHWARZ] >[PURPUR]'ZZZ'=ZU
RUECK ZUM MENUE"
16130 FOR I=1 TO 10
16140 PRINT "[REVERS EIN][BRAUN]
                                      ";:NEXT
16150 PRINT "[REVERS EIN]EINGABE
NR:-";SS+1:PRINT "[revers ein]";mid$(a$
(0,SS),2)
16160 PRINT "[REVERS EIN]";A$(1,SS):PRI
NT "[REVERS EIN]";D$(VAL(LEFT$(A$(0,SS)
,1)))
16170 Q1$="":INPUT "[SCHWARZ]BEFEHL:"
;Q1$
16180 IF Q1$<>"DDD" THEN 16210
16190 D(0,TEMP)=D(0,TEMP)-1:FOR I=SS TO
IT-1:A$(0,I)=A$(0,I+1)
16200 A$(1,I)=A$(1,I+1):NEXT I:IT=IT-1:G
OSUB 13160:GOTO 16040
```

```
16210 IF Q1$="ZZZ" THEN RETURN
16220 IF Q1$="" THEN SS=SS+1:GOTO 16040
16240 SS=SS+VAL(Q1$):IF SS<0 THEN SS=0
16250 GOTO16040
```

Ein direktes Benutzermodul für die Suche.

MODUL 5.1.8

```
17000 REM#******************************
17010 REM ZUFAELLIGE FRAGEN GENERIEREN
17020 REM*******************************
17030 QU=0
17040 PRINT "[ROT][RE
VERS EIN]FRAGEN"
17050 PRINT "[BLAU]MOECHTEN SIE, DASS
NUR ANTWORTEN AUS"
17060 INPUT "DEM GLEICHEN BEREICH WIE DI
E FRAGEN ER- SCHEINEN (J/N):";Q$
17065 PRINT ""
17070 IF Q$="J" THEN QU=1
17080 P1=0:P2=IT:Q1=INT(RND(0)*IT):Q2=IN
T(RND(0)*5):Q(Q2)=Q1
17090 IF QU=0 OR D(0,VAL(LEFT$(A$(0,Q1),
1)))<5 THEN 17110
17100 P1=D(1,VAL(LEFT$(A$(0,Q1),1))):P2=
D(0,VAL(LEFT$(A$(0,Q1),1)))
17110 FOR I=0 TO 4:IF I=Q2 THEN 17150
17120 PP=P1+INT(RND(0)*P2):IF PP=Q(Q2) T
HEN 17120
17130 FOR J=0 TO I:IF PP=Q(J) THEN 17120
17140 NEXT J:Q(I)=PP
17150 NEXT I
17160 PRINT "[BLAU]";NA$(1);":[GRUEN]
";A$(1,Q(Q2))
17170 PRINT "[PURPUR]";NA$(0);":[BR
AUN]"
17180 FOR I=0 TO 4:PRINT I+1;")  ";MID$(
A$(0,Q(I)),2):NEXT
17190 PRINT "[ROT]WELCHE IST IHRER MEIN
UNG NACH DIE RICH- TIGE ANTWORT ?"
```

```
17200 PRINT "▮[ORANGE]TIPPEN SIE DIE ENT
SPRECHENDE NUMMER EIN:":INPUT AA:QT=QT+1
17210 IF AA-1=Q2 THEN 17250
17220 PRINT "▮[SCHWARZ]FALSCH ! DIE RICH
TIGE ANTWORT IST:"
17230 PRINT MID$(A$(0,Q(Q2)),2):GOTO 172
70
17240 POKE 53281,0
17250 PRINT "▙[SCHWARZ]■■■■■■■■":FOR I=1 T
O 11:PRINT "                    RICHTIG!":NE
XT
17260 FOR I=1 TO 15:POKE 53281,I:FOR J=1
 TO 200:NEXT J,I:RR=RR+1
17270 PRINT "▮[BLAU][REVERS EIN]RETURN[R
EVERS AUS] FUER EINE NEUE FRAGE ODER 'ZZ
Z'"
17280 Q$="":INPUT "UM INS MENUE ZURUECKZ
UKEHREN:";Q$:IF Q$="ZZZ" THEN RETURN
17290 PRINT "▙";:GOTO 17080
```

Hier handelt es sich in Wirklichkeit um das einzige Originalmodul in dem Programm. Es soll eine zufällige Frage generieren, fünf mögliche Antworten auf dem Bildschirm anzeigen und eine Eingabe vom Benutzer akzeptieren, mit der die richtige Antwort - oder zumindest ein Versuch der richtigen Anwort - angegeben wird.

Kommentar

Zeilen 17050-17070: Mit dieser Routine wird die Schwierigkeit des Tests festgelegt. Mögliche Antworten können aus der gesamten Datei genommen werden. In diesem Fall sind viele der Antworten wahrscheinlich nicht richtig und machen die Auswahl der richtigen Antwort wesentlich einfacher. Alternativ können die Antworten nur aus derselben Typgruppe genommen werden, wodurch die Aufgabe wesentlich schwieriger wird, da sämtliche Antworten zumindest als möglich erscheinen.

Zeile 17080: Eine zufällige Frage wird aus der Datei genommen und eine zufällige Position für die Positionsnummer des Feldes Q gewählt.

Zeilen 17090-17100: Ist QU gleich Null oder stehen keine fünf Antworten in der Typgruppe, so entspricht der Bereich der möglichen Alternativantworten der ge-

samten Datei P1-P2. Ist QU gleich Eins und stehen mindestens fünf Antworten als richtige Antwort in derselben Gruppe, so werden P1 und P2 an den Anfang bzw. das Ende dieser Gruppe zurückgesetzt.

Zeilen 17110-17150: Vier zufällige Antworten werden aus dem Bereich gewählt, der von P1 und P2 festgelegt wird. Außerdem wird geprüft, ob die zufällig ausgewählte Anwort nicht der richtigen Antwort oder einer vorher ausgewählten zufälligen Antwort entspricht.

Zeilen 17160-17200: Die Frage wird zusammen mit den fünf möglichen Antworten ausgedruckt. Der Benutzer muß angeben, welche Antwort richtig ist. Die Variable QT, in der die Gesamtanzahl von Fragen festgehalten wird, wird erhöht.

Zeilen 17240-17260: Der Benutzer wird durch eine mehfach wechselnde Hintergrundfarbe belohnt. Die Variable RR wird erhöht.

Testen von Modul 5.1.8

Müssen einige Daten neu geladen werden, so sollte es nun möglich sein, eigene Tests mit mehreren Auswahlmöglichkeiten zu generieren. Der Benutzer kann die schwierigere Form des Tests hier nur generieren, wenn er ausreichend Daten eingegeben hat, mit denen das Programm ständig fünf Antworten desselben Typs anbieten kann.

MODUL 5.1.9

```
18000 REM#******************************
18010 REM PUNKTZAHL
18020 REM*******************************
18030 PRINT "[ROT][REVERS
EIN]PUNKTZAHL":IF QT=0 THEN RETURN
18040 PRINT "[GRUEN]ANZAHL FRAGEN
:";QT
18050 PRINT "RICHTIGE ANTWORTEN:";RR
18060 PRINT "[BLAU]PUNKTZAHL          :"
;INT(((RR-QT/5)/(QT*.8))*100);"%"
18070 PRINT"[SCHWARZ]MOECHTEN SIE DIE P
UNKTZAHL AUF 0 ZU-":INPUT"RUECKSETZEN (J
/N):";Q$
18080 IF Q$="J" THEN QT=0:RR=0
18090 RETURN
```

In diesem Modul wird die Punktzahl anhand der Tatsache berechnet, daß nur eine von fünf Antworten richtig sein kann, wenn die Antworten nur zufällig gewählt wurden.

Zusammenfassung

Dies ist ein recht leistungsfähiges Programm. Dies gilt jedoch nur, wenn der Benutzer genügend Daten eingibt, um dieses Programm voll ausschöpfen zu können. Aus diesem Programm wird wieder deutlich, daß der Benutzer, wenn er schon ein komplexes Programm schreibt, auch ein wenig weiter gehen und ein Mehrzweckprogramm schreiben kann, mit dem er sich in Zukunft sehr viel Arbeit spart.

Ein Schritt weiter

1. In seiner momentanen Form prüft das Programm, ob dieselbe Antwort nicht zwei Mal für eine Frage angezeigt wird. Es prüft jedoch nicht, ob zwei verschiedene Antworten in Wirklichkeit identisch sein könnten. Hier könnte eine entsprechende Prüfung eingesetzt werden.
2. Die Frage einer Belohnung bei erfolgreicher Ausführung ist besonders interessant - Erwachsene scheinen schon den Erfolg als Belohnung beim Spiel mit diesem Programm - oder besser: bei Benutzung desselben - zu betrachten. Bei Kindern sind jedoch alle Arten von Belohnung möglich. Wie wäre es mit einem kleinen Spiel in dem Programm, das nur drei Minuten lang ist oder nach richtiger Beantwortung einer Reihe von Fragen werden kann.

5.2 WORDS

Sobald man ein einwandfrei arbeitendes Programm hat, wird man feststellen, daß es auch anderweitig benutzt werden kann. So war es bei MultiQ. Das Ergebnis war das vorliegende Programm, das als spielerische Wort-Lernhilfe für Kinder benutzt werden kann, die erst mit dem Lesen beginnen. Der einzige wirkliche Unterschied zwischen diesem Programm und MultiQ besteht darin, daß die Fragen die Form von Bildern angezeigt werden und die Antworten Wörter darstellen, die zu den Bildern passen.
Was die Bilder selbst betrifft, so stammen sie aus einem anderen Programm in diesem Buch, nämlich Artist, das von Band gelesen und in das Verzeichnis dieses Programms geladen wurde. Die Kapazität des Programms, wie hier dargestellt, umfaßt 50 Bilder. Allerdings kann gegebenenfalls noch eine weitere Gruppe von Bildern von Band gelesen werden. Bilder, die von diesem Programm benutzt wer-

den sollen, dürfen nur die untersten zehn Zeilen des Bildschirms belegen, da der obere Teil des Bildschirms für die Fragen und Eingabeaufforderungen benötigt wird.

Words: Variablentabelle

A %	Speichert Daten für Bildzeichen und Farben.
B %	Koordinaten der Ecken des Bildes.
FNA(SS)	Wert des Elementes in A, dessen Position durch die Position der Ecken des Bildes bestimmt wird.
FNB(SS)	Tatsächlicher Zeichencode, der aus FNA abgeleitet wurde.
WW$	Eingabe der Antwort auf den Frage-Modul.

MODUL 5.2.1

```
11000 REM#******************************
11010 REM MENUE
11020 REM*******************************
11030 POKE 53281,15:PRINT "■■[BLAU]■■■■■
       [REVERS EIN]WORDS"
11040 PRINT "■MOEGLICHE BEFEHLE:"
11050 PRINT "■ [GRAU 2]1)NEUE OBJEKTE EI
NGEBEN"
11060 PRINT " 2)SUCHEN/LOESCHEN"
11070 PRINT " 3)FRAGEN GENERIEREN"
11080 PRINT " 4)PUNKTZAHL ANZEIGEN ODER
LOESCHEN"
11090 PRINT " 5)LADEN/SPEICHERN"
11100 PRINT " 6)INITIALISIEREN"
11110 PRINT " 7)BEENDEN"
11120 INPUT "■[HELLBLAU]BEFEHL:";Z:PRINT
  "■";
11130 ON Z GOSUB 13000,15000,16000,17000
,18000,12000,11140:GOTO 11000
11140 PRINT "■[BLAU]■■■■■■■■■■■■■■■■■■■■
        [REVERS EIN]E N D E":END
```

Ein Standard-Menümodul.

MODUL 5.2.2

```
12000 REM#******************************
12010 REM INITIALISIEREN
12020 REM******************************
12030 CLR:DIM Q(4),A$(49),A%(49,255),B%(
49,4):IT=0:R$=CHR$(13)
12040 GOTO 11000
12060 GOTO 11000
```

Initialisierung der Hauptvariablen.

MODUL 5.2.3

```
13000 REM#******************************
13010 REM NEUE OBJEKTE EINGEBEN
13020 REM******************************
13030 PRINT"[BLAU][REVERS E
IN]NEUE OBJEKTE"
13040 IF IT>=100 THEN PRINT "[ROT]KEIN
PLATZ MEHR.":FOR I=1 TO 2000:NEXT:RETURN

13050 INPUT "[BLAU]KASSETTE EINLEGEN, D
ANN [REVERS EIN]RETURN[REVERS AUS]:";Q$
13060 PRINT "":OPEN 1,1,0,"ARTIST":FOR
I=0 TO 4:INPUT#1,B%(IT,I):NEXT
13070 FOR I=0 TO B%(IT,4)-1:INPUT#1,C1,C
2
13080 A%(IT,I)=256*C1+C2-32767:NEXT I:CL
OSE1:SS=IT
13090 PRINT "";:GOSUB 14000
13100 INPUT "[BLAU]KORREKT (J/N):";Q$:I
F Q$="N" THEN RETURN
13110 INPUT "ZUM BILD PASSENDES WORT:";W
$
13120 INPUT "EINGABE RICHTIG (J/N):";Q$:
IF Q$="N" THEN 13090
13130 A$(IT)=W$:IT=IT+1
13140 INPUT "NOCH EIN BILD (J/N):";Q$:IF
 Q$="J" THEN 13000
13150 RETURN
```

Dieses Modul liest die von Artist erstellten Bilder von Band, so daß der Benutzer den Bildern das richtige Wort zuordnen kann.

MODUL 5.2.4

```
14000 REM#******************************
14010 REM BILD DARSTELLEN
14020 REM*******************************
14030 PP=0:FOR I=B%(SS,1)+1 TO B%(SS,3)-
1
14040 FOR J=B%(SS,0)+1 TO B%(SS,2)-1
14050 PP=PP+1:T1=A%(SS,PP)+32767:POKE 10
24+40*I+J,INT(T1/256)
14060 POKE55296+40*I+J,T1-256*INT(T1/256
):NEXT J,I
14070 RETURN
```

Dieses Modul benutzt die beiden definierten Funktionen, um die richtigen Zeichen und Farben aus den numerischen Werten auszugeben, die von Artist gesichert wurden, und diese mit POKE auf den Bildschirm und in den Farbspeicher zu setzen.

Kommentar

Zeile 14030: B%(SS,1) und B%(SS,3) zeichnen die vertikalen Koordinaten der oberen linken und unteren rechten Ecke des Bildes auf.

Zeile 14040: B%(SS,0) und B%(SS,2) zeichnen die horizontalen Koordinaten derselben Ecken auf.

Testen der Moduln 5.2.1–5.2.4

Der Benutzer sollte die von Artist erstellten Bilder laden können, sie auf dem Bildschirm ausdrucken und in die Hauptmatrix laden, wobei sie mit dem als Antwort ausgewählten Wort verknüpft werden.

MODUL 5.2.5

```
18000 REM#******************************
18010 REM LADEN/SPEICHERN
18020 REM*******************************
18030 PRINT "■[BLAU]KASSETTE EINLEGEN DA
```

```
NN [REVERS EIN]RETURN[REVERS AUS]--"
18040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
18050 PRINT "■[ROT]MOEGLICHE BEFEHLE:":P
RINT "[GRUEN] 1)DATEN SPEICHERN"
18055 PRINT "2)DATEN LADEN"
18060 INPUT "[SCHWARZ]BEFEHL:";Q:ON Q GO
TO 18070,18130:RETURN
18070 POKE 1,7:FOR I=1 TO 2000:NEXT
18080 OPEN1,1,1,"WORDS":PRINT#1,IT
18090 FOR I=0 TO IT-1:PRINT#1,A$(I);R$;B
%(I,4)
18100 FOR J=0 TO B%(I,4)-1:PRINT#1,A%(I,
J):NEXT J
18110 FOR J=0 TO 3:PRINT#1,B%(I,J):NEXT
J,I
18120 CLOSE1:RETURN
18130 OPEN1,1,0,"WORDS":INPUT#1,IT
18140 FOR I=0 TO IT-1:INPUT#1,A$(I),B%(I
,4)
18150 FOR J=0 TO B%(I,4)-1:INPUT#1,A%(I,
J):NEXT J
18160 FOR J=0 TO 3:INPUT#1,B%(I,J):NEXT
J,I
18170 CLOSE1:RETURN
```

Ein Standardmodul der Datendatei.

MODUL 5.2.6

```
15000 REM#*****************************
15010 REM SUCHEN/LOESCHEN
15020 REM******************************
15030 SS=0:IF IT=0 THEN RETURN
15040 PRINT "■■■■■■■■■■■■■■■■[ROT][REVERS
EIN]SUCHEN"
15050 IF SS>IT-1 THEN SS=IT-1
15060 IF SS<0 THEN SS=0
15070 PRINT "■■■■■■■■■■■■■■■■■■■■■■■■■■■[
GRUEN]ITEMS=";IT
15080 PRINT "[BLAU]MOEGLICHE BEFEHLE:"
```

```
15090 PRINT "█[SCHWARZ] >[PURPUR][REVERS
 EIN]RETURN[REVERS AUS] FUER NAECHSTES O
BJEKT."
15100 PRINT "[SCHWARZ] >[PURPUR]POS/NEG
NUMMER UM DEN ZAEHLER ZU VER-   AENDERN"

15110 PRINT "[SCHWARZ] >[PURPUR]'DDD' UM
 DAS OBJEKT ZU LOESCHEN"
15120 PRINT "[SCHWARZ] >[PURPUR]'ZZZ' UM
 INS MENUE ZURUECKZUKEHREN"
15130 PRINT "███[GRUEN]";A$(SS):GOSUB 14
000
15140 Q$="":INPUT "█[SCHWARZ]██████████BE
FEHL:";Q$
15150 IF Q$<>"DDD" THEN 15190
15160 FOR I=SS TO IT-1:FOR J=0 TO 255:A%
(I,J)=A%(I+1,J):NEXT J,I
15170 FOR I=SS TO IT-1:A$(I)=A$(I+1):FOR
J=0 TO 4:B%(I,J)=B%(I+1,J):NEXT J,I
15180 IT=IT-1:GOTO 15040
15190 IF Q$="ZZZ" THEN RETURN
15200 IF Q$="" THEN SS=SS+1:GOTO 15040
15210 SS=SS+VAL(Q$):GOTO 15040
15220 GOTO 15220
```

Ein einfaches Benutzermodul zur Suche.

MODUL 5.2.7

```
16000 REM#******************************
16010 REM ZUFAELLIGE FRAGEN
16020 REM*******************************
16030 Q1=INT(RND(0)*IT):Q2=INT(RND(0)*5)
:Q(Q2)=Q1
16040 FOR I=0 TO 4:IF I=Q2 THEN 16090
16050 PP=INT(RND(0)*IT)
16060 IF PP=Q1 THEN 16050
16070 FOR J=0 TO I:IF PP=Q(J) THEN16050
16080 NEXT J:Q(I)=PP
16090 NEXT I
16100 SS=Q1:PRINT "█":GOSUB 14000
```

```
16110 PRINT "■[GRUEN]";:FOR I=0 TO 4:PRI
NT "■";A$(Q(I)):NEXT
16120 PRINT "■[BLAU]TIPPEN SIE DAS RICHT
IGE WORT FUER DAS":INPUT"BILD:";WW$:QT=Q
T+1
16130 IF WW$<>A$(Q1) THEN PRINT "FALSCH!
 DIE ANTWORT IST [GRUEN]";A$(Q1):GOTO 16
160
16140 PRINT "■[SCHWARZ]RICHTIG [WEISS]RI
CHTIG [ROT]RICHTIG[CYAN] RICHTIG[PURPUR]
 RICHTIG":RR=RR+1
16150 FOR I=0 TO 15:POKE 53281,I:FOR J=1
 TO 150:NEXT J,I
16160 INPUT "[GRUEN]WEITER (J/N):";Q$:IF
 Q$="J" THEN 16000
16170 RETURN
```

Dieses Modul entspricht dem Generator für zufällige Fragen bei MultiQ, ist jedoch in sofern einfacher, als es stets mögliche Antworten aus der gesamten Datei mit Elementen wählt.

MODUL 5.2.8

```
17000 REM#*******************************
17010 REM PUNKTZAHL
17020 REM********************************
17030 IF QT=0 THEN RETURN
17040 PRINT "■■■■■■■■■■■■■■■■■■■■■■■[ROT][REVE
RS EIN]SCORE"
17050 PRINT "■■[GRUEN]ANZAHL FRAGEN    :"
;QT
17060 PRINT "■RICHTIGE ANTWORTEN:";RR
17070 PRINT "■[BLAU]PUNKTZAHL          :";
INT(((RR-QT/5)/(QT*.8))*100);"%"
17080 PRINT "■[SCHWARZ]MOECHTEN SIE DIE
PUNKTZAHL LOESCHEN":INPUT"(J/N):";Q$
17090 IF Q$="J" THEN QT=0:RR=0
17100 RETURN
```

Dieselbe Funktion wie das Modul für die Punktzahl in MultiQ.

Zusammenfassung

Auch hier handelt es sich um ein Programm, das etwas arbeitsaufwendig ist, wenn es von Nutzen sein soll, da das Erstellen der vielen kleinen Bilder für dieses Programm, einige Zeit in Anspruch nimmt. Eine einfache Antwort bestünde darin, sich mit anderen Commodore 64 Benutzern zusammenzutun und Bänder mit Bildern auszutauschen. Die Arbeit mit Mikrocomputern braucht niemanden vom Rest der Welt zu isolieren.

Ein Schritt weiter

Die Frage der Belohnung wird bei diesem Programm noch interessanter - hier sollte man sich vielleicht Gedanken machen, auf welche Weise eine richtige Antwort belohnt werden kann.

5.3 TYPIST

Nicht alle Lernfragen beziehen sich auf die Verarbeitung komplexer Daten. Ganz wichtige Lernvorgänge bestehen in der Einstudierung von Antworten. Und auf diesem Gebiet sind Computer ganz besonders leistungsfähig. Deshalb beginnen Piloten, Kapitäne und ähnliche Berufsgruppen ihre Ausbildung vor Simulatoren und nicht mit der teuren und gefahrvollen Realität. Dieses Programm kann nicht ganz mit einem Simulator verglichen werden, ist jedoch dennoch ein überaus leistungsfähiges Lernmittel.
Von allen Programmen, die ich geschrieben habe, ist dies eines meiner liebsten. Die Tatsache, daß es hier vorgestellt wird, beweist, daß ein Programm nicht lang zu sein braucht, um nützlich zu sein. Dieses Programm ist kurz, klar und gut in dem, was es vermittelt. Es hat mir übrigens dabei geholfen, mein Zwei-Finger-System beim Tippen zu verbessern. Von allen Versionen, die geschrieben wurden, ist die Version des C64 bei weitem die beste, so daß ich hoffe, daß Sie es in Ihre Programmsammlung aufnehmen werden.

Typist: Variablentabelle

C$	Zeile mit Leerzeichen, mit der Textzeilen gelöscht werden können.
CH	Anzahl von bis jetzt in den Tests vorhandenen Zeichen.
RIGHT	Anzahl richtiger Zeichen.
SUM	Anzahl eingegebener Zeichen.
TI	Systemvariable, die die abgelaufene Zeit in 60stel Sekunden zählt.
TT$	Temporärer Speicher für die in Anspruch genommene Zeit.

MODUL 5.3.1

```
11000 REM#******************************
11010 REM TASTATUR ZEICHNEN
11020 REM******************************
11030 POKE 53281,6
11040 PRINT "[SCHWARZ]
"
11050 A$="←1234567890+-£"
11060 PRINT "     ";:FOR I=1 TO 14:PRINT
"[REVERS EIN][SCHWARZ] [WEISS][REVERS AU
S]";MID$(A$,I,1);:NEXT
11070 PRINT "[REVERS EIN][SCHWARZ] [WEIS
S]C[SCHWARZ] [WEISS]D[SCHWARZ] [WEISS][R
EVERS AUS]"
11080 PRINT "     [SCHWARZ][REVERS EIN]
                                    "

11090 A$="QWERTYUIOP@*↑"
11100 PRINT "     [REVERS EIN] [WEISS]CC[
REVERS AUS]";:FOR I=1 TO LEN(A$):PRINT "
[SCHWARZ][REVERS EIN] [REVERS AUS][WEISS
]";MID$(A$,I,1);:NEXT
11110 PRINT "[REVERS EIN][SCHWARZ] [WEIS
S]RR[SCHWARZ] [WEISS][REVERS AUS]"
11120 PRINT "    [SCHWARZ][REVERS EIN]
                                    "

11130 A$="ASDFGHJKL:;="
11140 PRINT "    [REVERS EIN] [WEISS]R[SC
HWARZ]  [WEISS]S[REVERS AUS]";
11150 FOR I=1 TO LEN(A$):PRINT "[SCHWARZ
][REVERS EIN] [REVERS AUS][WEISS]";MID$(
A$,I,1);:NEXT
11160 PRINT "[REVERS EIN][SCHWARZ] [WEIS
S]RR[SCHWARZ] [WEISS][REVERS AUS]"
11170 A$="ZXCVBNM,./"
11180 PRINT "    [SCHWARZ][REVERS EIN]
                                    "

11190 PRINT "    [REVERS EIN][SCHWARZ] [W
EISS]R[SCHWARZ] [WEISS]SH[SCHWARZ] [REVE
RS AUS]";
```

```
11200 FOR I=1 TO LEN(A$):PRINT "[SCHWARZ
][REVERS EIN] [REVERS AUS][WEISS]";MID$(
A$,I,1);:NEXT
11210 PRINT "[REVERS EIN][SCHWARZ] [WEIS
S]SH[SCHWARZ] [WEISS]↑[SCHWARZ] [WEISS]←
[SCHWARZ] "
11220 PRINT "    [SCHWARZ][REVERS EIN]
                                      "

11230 PRINT "     [SCHWARZ][REVERS EIN]
      [WEISS]                  [SCHWARZ]
      "
```

Dieses Modul soll ganz einfach eine simple Kopie der Tastatur des C64 auf dem Bildschirm ausdrucken. Auf diese Weise kann der Benutzer den Bildschirm betrachten und braucht nicht auf die Tastatur zu sehen, wenn er eine Eingabe vornimmt.

Kommentar

Zeilen 11050-11070: Dieser Abschnitt druckt wie die folgenden Abschnitte die unregelmäßigen Enden der Tastatur aus (wie beispielsweise die RETURN- und RESTORE-Tasten), die das gleichmäßige Muster stören. Danach wird eine Zeile mit schwarzen inversen Leerzeichen ausgedruckt. Über diese Zeile werden dann die Namen der alphanumerischen Tasten an den entsprechenden Stellen gesetzt. Unter der Tastenzeile wird dann eine schwarze inverse Zeile angezeigt.

Testen von Modul 5.3.1

Mit dem Modul sollte eine Kopie der Tastatur in der oberen Hälfte des Bildschirms ausgedruckt werden.

MODUL 5.3.2

```
12000 REM#*****************************
12010 REM EINGABE
12020 REM******************************
12030 SUM=0:CH=0:RIGHT=0:RESTORE:TT$="00
0000"
12040 C$="
          "
12050 READ A$:IF A$="STOP" THEN GOTO 120
30
```

```
12060 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
12070 FOR LINE=1 TO 3:PRINT C$;:NEXT:PRI
NT "□□□□"
12080 IF LEN(A$)>39 THEN PRINT "STRING Z
U LANG":STOP
12090 PRINT "[SCHWARZ]";A$:PRINT "□[WEIS
S][REVERS EIN]";:FOR I=1 TO LEN(A$)
12100 GET T$:IF T$="" THEN 12100
12110 IF T$="□" OR T$="▒" OR T$="▌" OR T
$="▌" OR T$=CHR$(13) THEN GOTO 12100
12120 IF I=1 THEN TI$=TT$
12130 SUM=SUM+1:PRINT T$;" ▌";
12140 IF T$<>MID$(A$,I,1) THEN PRINT "←▌
";:GOTO 12100
12150 RIGHT=RIGHT+1:NEXT I:PRINT "[REVER
S AUS]":CH=CH+LEN(A$):TT$=TI$
12160 PRINT "▒▒▒▒";STR$(INT(RIGHT/SUM*10
00)/10);"%    "
12170 PRINT STR$(INT(SUM/(TI/6000))/100)
;" CPS    "
12180 INPUT "WEITER (J/N):";Q$
12190 POKE 780,0:POKE 781,21:POKE 782,0:
SYS 65520:PRINT C$
12200 IF Q$<>"N" THEN 12050
12210 END
```

Dieses Modul druckt eine zu kopierende Textzeile aus. Danach wird eine Taste für Taste vorgenommene Eingabe akzeptiert, wobei die benötigte Zeit und die Erfolgsrate verfolgt und Fehler angegeben werden.

Kommentar

Zeile 12050: Der zu kopierende Text wird in DATA-Anweisungen am Ende des Programms gespeichert. Diese DATA-Anweisungen müssen mit einer Zeile beendet werden, in der einfach STOP steht, wie in den angegebenen Beispielzeilen. Dadurch beginnt das Programm erneut mit dem Lesen.

Zeilen 12060-12090: Diese Zeilen benutzen die Folge mit Leerzeichen (C$), um den Bereich zu löschen, in dem Text ausgedruckt werden muß, den Text auszudrucken und die Druckposition nach unten zu verschieben, damit die Eingabe auf der nächsten Zeile aufgenommen werden kann.

Zeile 12110: Die Cursortasten werden nicht als Eingabe akzeptiert.

Zeile 12120: Das Programm verfolgt die Zeit, die für die Eingabe des Textes benötigt wird. Die Zeitzählung beginnt jedoch erst, nachdem der erste Buchstabe jeder Zeile eingegeben wurde, und wird zwischen den Zeilen eingestellt. Die ganze bislang in Anspruch genommene Zeit wird in TT$ gespeichert, auf das TI$ (siehe letztes Programm) am Anfang jeder Zeile gesetzt wird.

Zeilen 12130-1250: Der letzte eingegebene Buchstabe wird ausgedruckt. Ist er falsch, so wird dies durch eine Fehlermeldung angegeben und die Druckposition wieder auf diesen Punkt zurückgesetzt. Die Gesamtanzahl von betätigten Tasten und die Anzahl von richtig betätigten Tasten werden aufgezeichnet.

Zeilen 12160-12180: Bei Beendigung der Zeile wird die prozentuale Erfolgsrate angezeigt. Die Systemvariable TI, in der derselbe Wert wie in TI$ gespeichert wird, die jedoch in 60stel Sekunden ausgedrückt wird, wird zur Berechnung der Anzahl von pro Sekunde eingegebenen Zeichen benutzt. Diese scheinbar schwierige Formel gewährleistet, daß zwei Dezimalstellen normal ausgedruckt werden.

Zeile 12190: Diese Zeile zeigt eine andere Methode, mit der die Position festgelegt werden kann, bei der das nächste Zeichen ausgedruckt werden soll. Um diese Methode zu benutzen, muß Null mit POKE in Adresse 780, die Zeilenposition in Adresse 781 und die Spaltenposition in 782 gesetzt werden. Wird die ROM-Routine bei 65520 aufgerufen, so wird die Druckposition an diese Stelle bewegt. Mit dieser Methode können Zeichenfolgen ersetzt werden, die Cursor-Steuerzeichen enthalten. Hier wird mit dieser Methode einfach festgelegt, daß über die Eingabeaufforderung 'WEITER' eine Zeile mit Leerzeichen gedruckt wird, sobald das Programm fortgesetzt wird.

Testen von Modul 5.3.2

Dieses Modul kann erst getestet werden, nachdem Daten eingegeben wurden, die das Programm lesen kann. Zu diesem Zweck wird ein anderes Modul mit dem Übungstext bei Adresse 13000 eingegeben, das Modul mit einer Datenzeile beendet, in der STOP steht, und das Programm ausgeführt. Danach sollte die erste gespeicherte Textzeile angezeigt und wie in den Kommentaren beschrieben getestet werden.

BEISPIEL FÜR EINEN ÜBUNGSTEXT

```
13000 REM#*****************************
13010 REM DATEN FUER DIE TESTS
13020 REM******************************
13030 DATA "ASDF :LKJ ASDF :LKJ ASDF ;LK
J AS"
13040 DATA "ASDF :LKJ ASDF :LKJ ASDF ;LK
J AS"
13050 DATA "A AD ADD ADDS; ASK LAD ALL F
ALLS"
13060 DATA "A AD ADD ADDS; ASK LAD ALL F
ALLS"
13070 DATA "A AD ADD ADDS; ASK LAD ALL F
ALLS"
13080 DATA STOP
```

Zusammenfassung

Hier handelt es sich um ein Programm, das nur eingesetzt werden kann, wenn es ernsthaft benutzt wird. Am besten nutzt man dieses Programm, indem man ein Buch mit Schreibübungen nimmt und dieses Buch als Grundlage für die eingegebenen Daten benutzt. Nachdem Sie sich dieser Mühe unterzogen haben, werden Sie feststellen, daß es sich hier um ein wirksames Hilfsmittel handelt, mit dem die Fähigkeiten des Maschineschreibens wesentlich verbessert werden.

Ein Schritt weiter

Die richtige Technik beim Maschineschreiben hängt davon ab, ob der richtige Finger für die jeweilige Taste benutzt wird. Bei einem derartigen Lernmittel sollten einfach die Tasten auf der Tastatur farbig merkiert werden, um anzugeben, welcher Finger benutzt werden soll. Schließlich wäre es eine gute Ubung, Farbeigenschaften innerhalb einer Zeichenfolge zu benutzen.

KAPITEL 6

HOCHENTWICKELTE FINANZFUNKTIONEN MIT MIKROCOMPUTERN

Trotz der Witze über die Gasrechnungen in Höhe von einer Million DM können Computer Finanzinformationen besonders gut verarbeiten. Dies ist nicht nur auf die Tatsache zurückzuführen, daß sie die Daten so viel schneller als ein Mensch speichern und verarbeiten können, sondern auch auf ihre Fähigkeit, die Fakten auf klare und verständliche Weise darzulegen. In diesem Kapitel werden drei Finanzprogramme vorgestellt, die die Rechenmöglichkeiten des Commodore 64 und seine flexiblen Funktionen zur Bildschirmverarbeitung benutzen, um den Zahlen etwas von ihrem Geheimnis zu nehmen.

6.1 BANKER

Unser erstes Programm ist Banker, ein einfaches Hilfsmittel, mit dem der Status eines Bankkontos verfolgt werden kann, bevor der gefürchtete Umschlag von der Bank im Briefkasten liegt. Das Programm befaßt sich mit Abbuchungen und Eingängen, Daueraufträgen und Einzelpositionen, wobei ein klar formatierter Auszug für jeden gewünschten Monat erzeugt wird. Im Laufe des Programms werden Sie auf einige der Probleme stoßen, die bei der Aufstellung numerischer Daten in verständlicher Form auf dem Bildschirm auftreten.

Banker: Variablentabelle

A(99,1)	Speicherung der Zahlungsbeträge und des Zahlungsdatums.
A$(99,1)	Speicherung der Zahlungsnamen und Monate, in denen die Zahlung vorgenommen werden muß.
CD	Flag, mit dem angegeben wird, ob es sich um eine Haben- oder Sollzahlung handelt.
CR$	Separator für Datendateien.
IN	Initialisierungs-Flag.
M	Monats-Nummer minus 1.
MM	Temporäre Variable, die bei der Formatierung von Zahlungsbeträgen benutzt wird.
MM$	Wird zur Aufnahme des formatierten Zahlungsbetrages benutzt.
MO$	Speicher der Monatsnamen.
PA	Anzahl gespeicherter Zahlungen.

R$	Temporäre Zeichenfolge, in der die Monate gespeichert werden, in denen eine Zahlung vorgenommen werden muß.
S	Temporärer Speicher für den Zahlungstag in dem angegebenen Monat.
SUM	Wird zur Kumulierung von Beträgen für die Gesamtsumme in dem Kontoauszug benutzt.

MODUL 6.1.1

```
11000 REM#******************************
11010 REM MENUE
11020 REM*******************************
11030 POKE 53281,7:PRINT "
    [REVERS EIN][ROT]BANKER[REVERS EIN][
BLAU]"
11040 PRINT "MOEGLICHE BEFEHLE:"
11050 PRINT "   1)NEUE ZAHLUNGEN"
11060 PRINT "   2)AENDERN/LOESCHEN VON
DATEN"
11070 PRINT "   3)KONTOAUSZUG AUSGEBEN"

11080 PRINT "   4)LADEN/SPEICHERN"
11090 PRINT "   5)INITIALISIEREN"
11100 PRINT "   6)BEENDEN"
11110 INPUT "[SCHWARZ]BEFEHL:";Z:PRINT
 "";
11120 IF Z=5 OR Z=6 OR IN=1 THEN 11140
11130 PRINT"NOCH NICHT I
NITIALISIERT."
11135 FOR I=1 TO 2000:NEXT:GOTO 11000
11140 IF PA<>0 OR (Z<>2 AND Z<>3) THEN 1
1160
11150 PRINT "NOCH KEINE DAT
EN VORHANDEN."
11155 FOR I=1 TO 2000:NEXT:GOTO 11000
11160 ON Z GOSUB 13000,14000,15000,16000
,12000,11180
11170 GOTO 11000
11180 PRINT "
[REVERS EIN][ROT]BANKER"
11190 PRINT "[REVERS AUS][SCH
```

```
WARZ]PROGRAMM BEENDET"
11200 END
```

Ein Standard-Menümodul.

MODUL 6.1.2

```
12000 REM#******************************
12010 REM VARIABLEN INITIALISIEREN
12020 REM*******************************
12040 CLR:IN=1:DIM A$(99,1),A(99,1):A(0,
1)=999
12050 RESTORE
12060 S$="        "
12070 DIM MO$(11):FOR I=0 TO 11:READ A$:
MO$(I)=A$:NEXT:CR$=CHR$(13)
12080 DATA JANUAR,FEBRUAR,MAERZ,APRIL,MA
I,JUNI,JULI,AUGUST,SEPTEMBER,OKTOBER
12090 DATA NOVEMBER,DEZEMBER
12100 GOTO 11000
```

Initialisiert Variablen und setzt dann die Monatsnamen in MO$.

MODUL 6.1.3

```
13000 REM#******************************
13010 REM NEUE ZAHLUNGEN EINGEBEN
13020 REM*******************************
13030 PRINT "[ROT][REVERS EIN]
   NEUE ZAHLUNGEN[BLAU][REVERS AUS]"
13040 PRINT "  1)HABEN":PRINT "  2)SOLL"

13050 INPUT "[SCHWARZ]EINGABE:";CD:CD=C
D-1
13060 INPUT "[BLAU]NAME DER ZAHLUNG:";
Q$
13070 INPUT "BETRAG:";Q
13080 INPUT "MONATE (Z.B. 01040710):";R
$:PRINT "";
13090 FOR I=1 TO LEN(R$) STEP 2:LET M=VA
L(MID$(R$,I,2))-1
13100 IF M>=0 AND M<=11 THEN GOTO 13130
```

```
13110 PRINT:PRINT "□UNGUELTIGE MONATSEIN
GABE                     "
13120 FOR J=1 TO 2000:NEXT:GOTO 13080
13130 PRINT MO$(M);"/";:NEXT:PRINT
13140 INPUT "▓TAG DER ZAHLUNG:";S
13150 INPUT "▓▓[ROT]EINGABE KORREKT (J/N
):";T$:IF T$="N" THEN PRINT"";:GOTO 130
00
13160 PA=PA+1:FOR J=PA-1 TO 0 STEP -1
13170 IF S<A(J,1) THEN FOR K=0 TO 1:A$(J
+1,K)=A$(J,K):A(J+1,K)=A(J,K):NEXT K,J
13180 J=J+1:A$(J,1)="000000000000"
13190 FOR I=1 TO LEN(R$) STEP 2:M=VAL(MI
D$(R$,I,2))
13200 A$(J,1)=LEFT$(A$(J,1),M-1)+"1"+RIG
HT$(A$(J,1),12-M):NEXT
13210 A$(J,0)=Q$:A(J,0)=Q:A(J,1)=S
13220 IF CD=1 THEN A(J,0)=A(J,0)*-1
13230 RETURN
```

Dieses Modul ermöglicht die Eingabe der Zahlungsnamen und der zugehörigen Daten.

Kommentar

Zeilen 13080-13130: Die Monate, in denen eine Zahlung vorgenommen wird, können zwischen 1 für eine Einzelzahlung und 12 für einen Dauerauftrag variieren. Die Monate werden in Form einer Zeichenfolge aus zweistelligen Zahlen eingegeben, die gelesen und geprüft werden. Danach werden die Monatsnamen auf dem Bildschirm zur Überprüfung ausgedruckt. Mit dieser einfachen Eingabemethode wird ein hoher Flexibilitätsgrad ohne komplexe Programmierung erzielt.

Zeile 13170: Die Zahlungen werden in einer einzigen Matrix je nach Zahlungsdatum gespeichert. Die Einfügung wird erzielt, indem die Datei einfach vom höchsten Tag an durchsucht wird.

Zeilen 13180-13200: Diese Zeilen legen einen Monatsindikator fest, der aus zwölf Nullen besteht. Die angegebenen Monate werden dann durchsucht, und die entsprechenden Stellen in dem Indikator werden auf 1 gesetzt.

Zeile 13220: Die Abbuchungen und Eingänge werden beide als positive Summen eingegeben. Wird mit der Variablen CD ein Soll angegeben, so wird der Betrag mit minus 1 multipliziert.

MODUL 6.1.4

```
14000 REM#*****************************
14010 REM AENDERN/LOESCHEN VON DATEN
14020 REM******************************
14030 FOR I=0 TO PA-1:PRINT "";
14040 PRINT "ZAHLUNG:";A$(I,0)
14050 PRINT "BETRAG:";A(I,0)
14060 PRINT "MONATE:";:FOR J=1 TO 12
14070 IF MID$(A$(I,1),J,1)="1" THEN PRIN
T MO$(J-1);"/";
14080 NEXT J:PRINT
14090 PRINT "TAG DER ZAHLUNG:";A(I,1)
14100 PRINT "[ROT][REVERS EIN]FUNKTIO
NSTASTEN-BEFEHLE:[REVERS AUS][BLAU]":PRI
NT "1 - NAECHSTE ZAHLUNG":
14110 PRINT "2 - ZUM MENUE":PRINT "3 -
 LOESCHEN"
14120 PRINT "[GRUEN]BEFEHL:?[SCHWARZ]"
14130 GET Q$:IF Q$="" THEN 14130
14140 IF ASC(Q$)<>140 THEN 14170
14150 FOR J=I TO PA-1:FOR K=0 TO 1:A$(J,
K)=A$(J+1,K):A(J,K)=A(J+1,K):NEXT K,J
14160 PA=PA-1:RETURN
14170 IF ASC(Q$)=133 THEN NEXT I
14180 RETURN
```

Ein einfaches Benutzer-Suchmodul, mit dem die Zahlungen und die Monate ausgedruckt werden, in denen die Zahlungen vorgenommen werden müssen. Außerdem können Löschungen mit drei der Funktionstasten zur Eingabe von Befehlen vorgenommen werden.

Testen der Moduln 6.1.1 - 6.1.4

Nun sollte der Benutzer Zahlungen und die Monate eingeben können, in denen die Zahlungen vorgenommen werden müssen. Außerdem sollte er diese Angaben nach Belieben durchsuchen und löschen können.

MODUL 6.1.5

```
16000 REM#*****************************
16010 REM LADEN/SPEICHERN
```

```
16020 REM******************************
16030 PRINT "KASSETTE EINLEGEN, DANN [RE
VERS EIN]RETURN[REVERS AUS] "
16040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$
16060 POKE 192,7:POKE 1,39
16070 PRINT "[ROT]■MOEGLICHE FUNKTIONEN:
":PRINT "[GRUEN]1)daten speichern"
16075 PRINT "■2)DATEN LADEN"
16080 INPUT "■[BLAU]EINGABE:";Q:ON Q GOT
O 16100,16150
16090 RETURN
16100 POKE 192,0:FOR I=1 TO 5000:NEXT
16110 OPEN 1,1,2,"BANKER"
16120 PRINT#1,PA
16130 FOR I=0 TO PA-1:PRINT#1,A$(I,0),CR
$,A$(I,1),CR$,A(I,0),CR$,A(I,1):NEXT
16140 CLOSE1:RETURN
16150 OPEN 1,1,0,"BANKER"
16160 INPUT#1,PA
16170 FOR I=0 TO PA-1:INPUT#1,A$(I,0),A$
(I,1),A(I,0),A(I,1):NEXT
16175 A(PA,1)=999
16180 CLOSE1
16190 GOTO 11000
```

Ein Standardmodul einer Datendatei.

MODUL 6.1.6

```
17000 REM#*****************************
17010 REM FORMATTIEREN DER BETRAEGE
17020 REM******************************
17030 MM=MM+10000.001
17040 M$=MID$(STR$(MM),3,LEN(STR$(MM))-3
)
17050 FOR FF=1 TO 7:IF MID$(M$,FF,1)<>"0
" THEN RETURN
17060 M$=LEFT$(M$,FF-1)+" "+RIGHT$(M$,7-
FF):NEXT FF
```

Dieses kurze Modul stellt eine einfache Methode zur Erzielung eines Standardformats für die Geldbeträge dar, die von dem nächsten Modul ausgedruckt werden müssen.

Kommentar

Zeile 17030: Dieses Modul wird von vielen verschiedenen Stellen in dem nächsten Modul des Programms aufgerufen und bearbeitet Werte aus verschiedenen Variablen. Um dies zu erreichen, muß der zu formatierende Wert zuerst in der Variablen MM gespeichert werden. Der erste Schritt des Formatierverfahrens besteht darin, daß 10000.001 zu dem Betrag hinzugefügt wird, wobei davon ausgegangen wird, daß der Betrag nicht größer ist als 9999.99. Dadurch ergibt sich eine Standardlänge von neun Zeichen (einschließlich dem Dezimalpunkt), von denen das erste und letzte Zeichen redundant sind.

Zeilen 17040: Der Wert wird nun in eine Zeichenfolge umgesetzt, wobei das erste und letzte Zeichen (die 1) abgeschnitten werden. Wird eine Zahl mit der Funktion STR$ in eine Zeichenfolge umgesetzt, so wird automatisch ein Leerzeichen vorangestellt, so daß das zweite Zeichen der Zahl in Wirklichkeit in Position 3 der Zeichenfolge steht.

Zeilen 17050-17060: In diesem Programm sind keine führenden Nullen erforderlich, so daß diese Routine die Zeichenfolgen durchsucht und dabei führende Nullen in Leerzeichen umsetzt. Daraus ergibt sich eine Zeichenfolge aus jeweils sieben Zeichen, einschließlich zwei Dezimalstellen. Derartige Zeichenfolgen können in Spalten ausgedruckt werden, wobei man sicher sein kann, daß die Position der Dezimalpunkte stets übereinstimmt.

MODUL 6.1.7

```
15000 REM#*****************************
15005 REM KONTOAUSZUG
15010 REM******************************
15020 PRINT "███████████████[ROT]KONTOAUSZ
UG[SCHWARZ]":SUM=0
15030 INPUT "█NUMMER DES MONATS FUER KON
TOAUSZUG:";Q
15040 IF Q=1 THEN 15080
15050 FOR Q1=1 TO Q-1:FOR I=0 TO PA-1:IF
 MID$(A$(I,1),Q1,1)<>"1" THEN 15070
15060 SUM=SUM+A(I,0)
15070 NEXT I,Q1
```

```
15080 PRINT "[ROT]";MO$(Q-1)
15090 PRINT "[GRUEN][REVERS EIN]
                    ZAHLUNG  TOTAL[REVERS AU
S]"
15100 PRINT "[SCHWARZ]SALDO
                  ";
15110 IF SUM<0 THEN PRINT "[ROT]";
15120 LET MM=ABS(SUM):GOSUB 17000:PRINT
M$
15130 FOR I=0 TO PA-1
15140 IF MID$(A$(I,1),Q,1)<>"1" THEN 152
20
15150 LET MM=A(I,1):GOSUB 17000:PRINT "[
SCHWARZ]";MID$(M$,2,3);:PRINT "[SCHWARZ]
   ";
15160 IF A(I,0)<0 THEN PRINT "[ROT]";
15170 PRINT A$(I,0)
15180 PRINT "";:
MM=ABS(A(I,0)):GOSUB 17000:PRINT M$;
15190 SUM=SUM+A(I,0):PRINT "[SCHWARZ]"
;:IF SUM<0 THEN PRINT "[ROT]";
15200 MM=ABS(SUM):GOSUB 17000:PRINT M$
15210 GET A$:IF A$="" THEN 15210
15220 NEXT I:INPUT "ZURUECK MIT [REVERS
EIN]RETURN[REVERS AUS]:";X
15230 RETURN
```

Dieses Modul erzeugt eindeutig formatierte Monatsauszüge - und erfüllt somit den Zweck des Programms.

Kommentar

Zeilen 15030-15070: Die Monatsindikatoren sämtlicher gespeicherten Zahlungen werden durchsucht, um festzustellen, ob Zahlungen unter den Titeln für Monate vor dem Datum des Kontoauszugs vorgenommen wurden. Derartige Zahlungen werden kumuliert, um einen Saldo am Anfang des angegebenen Monats zu erhalten.

Zeilen 15100-15120: Der Saldo wird ausgedruckt, wobei er mit dem vorhergehenden Modul formatiert wird. Hier wird darauf hingewiesen, wie einfach ein negativer Saldo durch Ausdrucken des roten Steuerzeichens angegeben werden kann.

Zeilen 15130-15220: Die Datei mit den Zahlungen wird nun nach den Zahlungen für den laufenden Monat durchsucht. Wann immer eine relevante Zahlung gefunden wird, wird der Tag auf der linken Seite des Bildschirms ausgedruckt, wobei das Druckformat mit dem Formatmodul standardisiert wird, die hinzugefügten Dezimalpunkte jedoch abeschnitten werden. Danach wird der Name der Zahlung ausgedruckt, gefolgt von dem Betrag, wobei das rote Steuerzeichen hinzugefügt wird, wenn es sich um ein Soll handelt. Schließlich wird die Zahlung zu der Variablen SUM hinzugefügt und der aktuelle Saldo wird neben dem Betrag der Zahlung ausgedruckt, wieder in rot, wenn der Saldo negativ ist. Anhand der flexiblen Cursorsteuerung des C64 ist das Erstellen derartiger Tabellen problemlos - weist die Tabelle nicht das richtige Format auf, so wird eine Cursorbewegung mehr oder weniger ausgeführt, bis die Tabelle das richtige Format aufweist.

Zeile 15210: Die Zahlungen werden nacheinander ausgedruckt, wobei die nächste Zahlung erst nach Betätigung einer beliebigen Taste ausgedruckt wird. Dadurch wird verhindert, daß Zahlungen vom oberen Bildschirmrand laufen, bevor sie überprüft werden können.

Testen der Moduln 6.1.6 - 6.1.7

Wurden einige Daten gespeichert, so sollten diese nun wieder in den C64 zurückgeladen und ein monatlicher Kontoauszug aufgerufen werden, wobei die Beträge und Zahlungstitel sauber in Spalten formatiert sind. Ist der monatliche Kontoauszug richtig formatiert, so ist das Programm einsatzbereit.

Zusammenfassung

Dieses sehr direkte Programm erhebt einige interessante Fragen darüber, wie komplex ein Programm sein muß, um nützlich zu sein. Die Eingabe der Monate in Form einer Zeichenfolge ist in vielerlei Hinsicht recht einfach im Vergleich zu der Angabe, ob die Zahlung monatlich, vierteljährlich oder jährlich vorgenommen werden muß, wobei das Programm dann die Zahlung in die entsprechenden Monate einfügt. Eine derartige zusätzliche Funktion wäre problemlos möglich, würde jedoch die Länge des Programms vergrößern und die Flexibilität der direkten Angabe von Monaten reduzieren, wodurch sogar unregelmäßige Monate eingegeben werden können. Schreibt der Benutzer eigene Programme, so wird er ständig mit der Frage konfrontiert, was besser automatisch und was besser vom Benutzer getan werden soll. Die Antwort kann von Benutzer zu Benutzer durchaus verschieden sein, aber Komplexität nur um der Komplexität willen kann sehr teuer werden, was den Speicherplatz betrifft, und gelegentlich sogar die Nützlichkeit eines Programmes einschränken.

Ein Schritt weiter

1. Das Löschmodul ist insofern recht einfach, als es dem Benutzer nur ermöglicht, die Einträge nacheinander durchzusehen. Warum keine Funktion hinzufügen, mit der ein positiver oder negativer Sprung angegeben wird, wobei eines der vorhergehenden Programme als Beispiel benutzt werden kann.
2. Eine weitere Verbesserung bestünde darin, ein binäres Suchmodul hinzuzufügen, mit dem das momentan benutzte Durchsuchen von Ende der Datei beim Einfügen von Elementen ersetzt wird.
3. Die Monatsindikatoren benutzen insgesamt 12 Bytes für jede Zahlung. Anhand der Angaben über AND und OR sollte der Benutzer nun in der Lage sein, dieselben Informationen mit 2 Bytes zu speichern und zurückzuübertragen (d.h. mit einem Element in einem ganzzahligen Feld).

6.2 ACCOUNTANT

Dieses Programm führt nicht wirklich ihre die Bücher, sondern gestaltet nur die Buchführung einfacher und stellt die einzelnen Zahlungen gegebenenfalls in einem geordneten Format dar, wobei einzelne Zahlungen, Haupt- und Untertitel beim Ausdrucken der einzelnen Konten entsprechend berücksichtigt werden.

Accountant: Variablentabelle

A$(1,99)	Hauptdatei mit den Zahlungsnamen.
A(1,99)	Hauptdatei mit den Zahlungsbeträgen.
C$	Zeile mit Leerzeichen, die beim Löschen von Texten benutzt wird.
C(1)	Feld, in dem die Anzahl von Zahlungen auf der Haben-/Sollseite der Konten gespeichert wird.
CD	Indikator, mit dem angegeben wird, ob es sich um eine Haben- oder Soll-Zahlung handelt.
CR$	Separator der Datendatei.
GR	Wird dazu benutzt, die Anzahl von Zahlungen unter einem einzigen Haupttitel aufzuzeichnen.
HH$	Temporärer Speicher für den Namen des Haupttitels in dem Benutzer-Suchmodul.
IN	Indikator für die Initialisierung.
M$	Zeichenfolge, in der formatierte Geldbeträge gespeichert werden.
MM	Temporäre Variable, die beim Formatieren der Geldbeträge benutzt wird.
PL	Platz in der Datei für das Einfügen eines neuen Untertitels.

SS	Temporäre Variable, mit den Zahlungen unter einem Haupttitel kumuliert werden.
TT	Wird zur Kumulierung von Zahlungen in Konten benutzt.

MODUL 6.2.1

```
11000 REM#********************************
11010 REM MENUE
11020 REM*********************************
11030 POKE 53281,7:PRINT "[BLAU][REVER
S EIN]                 ACCOUNTANT[REVERS AUS
]"
11040 PRINT "[ROT]MOEGLICHE BEFEHLE:"
11050 PRINT "[GRUEN]   1)NEUE TITEL EI
NGEBEN"
11060 PRINT "    2)AENDERN/LOESCHEN VON
DATEN"
11070 PRINT "    3)RECHNUNG AUSGEBEN"
11080 PRINT "    4)LADEN/SPEICHERN
11090 PRINT "    5)INITIALISIEREN"
11100 PRINT "    6)BEENDEN"
11110 INPUT "[BLAU]BEFEHL:";Z:PRINT "
";
11120 IF Z<1 OR Z>4 OR IN=1 THEN 11140
11130 PRINT "NOCH NICHT
INITIALISIERT"
11135 FOR I=1 TO 2000:NEXT:GOTO 11000
11140 IF Z<4 AND Z>0 THEN GOSUB 13000:PR
INT "";
11150 ON Z GOSUB 14000,17000,19000,20000
,12000,11170:Z=0:GOTO 11000
11160 .
11170 PRINT "[R
OT][REVERS EIN]ACCOUNTANT[REVERS AUS]"
11180 PRINT "[BLAU]PROGRAMM
BEENDET":END
```

Standard-Menümodul.

MODUL 6.2.2

```
12000 REM#*****************************
12010 REM INITIALISIEREN
12020 REM******************************
12030 CLR:DIM A$(1,33),A(1,33):CR$=CHR$(
13)
12040 C$="
         "
12050 IN=1:GOTO 11000
```

Initialisierungsmodul.

MODUL 6.2.3

```
13000 REM#*****************************
13010 REM SOLL ODER HABEN
13020 REM******************************
13030 PRINT "1)HABE
N2)soll"
13040 INPUT "EINGABE:";CD:
CD=CD-1:RETURN
```

Vor der Eingabe eines Postens muß der Benutzer angeben, ob es sich um einen Haben- oder Sollzahlung handelt.

MODUL 6.2.4

```
14000 REM#*****************************
14010 REM TITEL EINGEBEN
14020 REM******************************
14030 PRINT "[ROT][REVERS
 EIN]NEUE DATEN:";:IF CD=0 THEN PRINT "H
ABEN"
14040 IF CD=1 THEN PRINT "SOLL"
14050 PRINT "[REVERS AUS][BLAU]IST ES:"
:PRINT "[GRUEN]  1)EINE EINZELNE ZAHLUN
G"
14060 PRINT "  2)EIN NEUER HAUPTTITEL":
PRINT "  3)EIN UNTERTITEL"
14070 PRINT "  0) UM INS MENUE ZURUECKZ
UKEHREN"
```

```
14080 INPUT "■[PURPUR]EINGABE:";TYPE
14090 ON TYPE GOTO 15000,15000,16000
14100 RETURN
```

Bei der Eingabe einer Zahlung muß der Benutzer angeben, ob es sich um einen Haupttitel, einen Untertitel oder eine einzelne Zahlung handelt. Werden die Konten ausgedruckt, so werden bei den Haupttitel keine Summen angegeben, die Untertitel unter die entsprechenden Hauptteil gesetzt und für die Gruppe eine Zwischensumme ausgedruckt. Einzelne Zahlungen werden allein mit einer entsprechenden Summe angegeben.

MODUL 6.2.5

```
15000 REM#******************************
15010 REM EINZELNE ZAHLUNG ODER TITEL
15020 REM******************************
15030 Q=0:INPUT "■[BLAU]NAME DER ZAHLUNG
:";Q$
15040 IF TYPE<>2 THEN INPUT "■BETRAG FUE
R DIE ZAHLUNG:";Q
15050 INPUT "■[ROT]EINGABE KORREKT (J/N)
:";R$:IF R$="N" THEN 14000
15060 Q$="%"+Q$:IF TYPE=2 THEN Q$="*"+MI
D$(Q$,2)
15070 A$(CD,C(CD))=Q$:A(CD,C(CD))=Q:C(CD
)=C(CD)+1:GOTO 14000
```

In dieses Modul werden die Haupttitel oder einzelne Zahlungen eingegeben.

Kommentar

Zeile 15040: Ein Betrag wird nur angefordert, wenn es sich bei der Zahlung nicht um einen Haupttitel handelt.

Zeile 15060: An den Anfang der Zahlung wird ein Indikator angefügt- für eine einzelne Zahlung, * für einen Haupttitel. Diese Indikatoren werden niemals ausgedruckt, sondern vom Programm zur Kennzeichnung der verschiedenen Typen benutzt.

Zeile 15070: Hier wird auf die Variable CD verwiesen, mit der angegeben wird, auf welcher Seite der Hauptfelder Name und Betrag gespeichert werden.

MODUL 6.2.6

```
16000 REM#*****************************
16010 REM UNTERTITEL
16020 REM******************************
16030 INPUT "■[BLAU]NAME DES HAUPTTITELS
:";Q$:Q$="*"+Q$
16040 FOR I=0 TO C(CD)-1:IF A$(CD,I)=Q$
THEN 16060
16050 NEXT:PRINT "HAUPTTITEL NICHT GEFUN
DEN":FOR I=1 TO 2000:NEXT:GOTO 14000
16060 PL=I+1:INPUT "■[SCHWARZ]NAME DES U
NTERTITELS:";Q$
16070 INPUT "■BETRAG:";Q
16080 INPUT "■[ROT]EINGABE KORREKT (J/N)
:";R$:IF R$="N" THEN GOTO 14000
16090 Q$="$"+Q$
16100 FOR I=C(CD)+1 TO PL+1 STEP -1:A$(C
D,I)=A$(CD,I-1):A(CD,I)=A(CD,I-1):NEXT
16110 A$(CD,PL)=Q$:A(CD,PL)=Q:C(CD)=C(CD
)+1:GOTO 14000
```

In dieses Modul werden die Untertitel eingegeben.

Kommentar

Zeilen 16030-16050: Der Name des entsprechenden Haupttitels wird angefordert und mit den Titeln in der Datei verglichen. Ist der Haupttitel nicht vorhanden, so wird eine Fehlermeldung generiert. Hier wird noch einmal darauf hingewiesen, wie eine Schleife für eine Suche benutzt wird, wobei das Programm aus der Schleife geht, wenn die Zahlung gefunden wurde und der Wert der Schleifenvariablen I dazu benutzt wird, die Stelle festzulegen, an der der Untertitel eingefügt wird. Die Beendigung der Schleife bedeutet, daß der Haupttitel nicht vorhanden ist.

Zeilen 16090-16110: Der Untertitel wird mit einem Symbol gekennzeichnet und unmittelbar im Anschluß an den entsprechenden Haupttiel zu der Hauptdatei hinzugefügt.

Testen der Moduln 6.2.1 - 6.2.6

Nun sollte der Benutzer in der Lage sein, Haben- oder Soll-Zahlungen in das Kon-

to einzugeben und diese auf der richtigen Seite der Hauptdatei einzufügen (Habenseite 0, Sollseite 1). Dies kann nur im direkten Modus überprüft werden.

MODUL 6.2.7

```
20000 REM#******************************
20010 REM LADEN/SPEICHERN
20020 REM*******************************
20030 PRINT "█[BLAU]KASSETTE EINLEGEN, D
ANN [REVERS EIN]RETURN[REVERS AUS]---"
20040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
20050 PRINT "██[ROT]MOEGLICHE BEFEHLE:":
PRINT "[GRUEN]█1)DATEN SPEICHERN"
20055 PRINT "█2)DATEN LADEN"
20060 INPUT "█[ROT]BEFEHL:";Q:ON Q GOTO
20080,20140
20070 RETURN
20080 FOR I=0 TO 1:IF A$(I,0)="" THEN A$
(I,0)=" ":NEXT
20090 POKE 1,7:FOR I=1 TO 2000:NEXT
20100 OPEN 1,1,2,"RECHNUNG"
20110 FOR I=0 TO 1:PRINT#1,C(I):IF C(I)=
0 THEN 20130
20120 FOR J=0 TO C(I)-1:PRINT#1,A$(I,J),
CR$,A(I,J):NEXT J
20130 NEXTI:CLOSE1:RETURN
20140 OPEN 1,1,0,"RECHNUNG"
20150 FOR I=0 TO 1:INPUT#1,C(I):IF C(I)=
0 THEN 20170
20160 FOR J=0 TO C(I)-1:INPUT#1,A$(I,J),
A(I,J):NEXT J
20170 NEXT I:CLOSE1:RETURN
```

Ein Standardmodul der Datendatei.

MODUL 6.2.8

```
21000 REM#******************************
21010 REM FORMATTIERUNG DER BETRAEGE
21020 REM*******************************
```

```
21030 MM=MM+10000.001:M$=MID$(STR$(MM),3
,7)
21040 FOR P=1 TO 3
21050 IF MID$(M$,P,1)="0" THEN M$=LEFT$(
M$,P-1)+" "+RIGHT$(M$,7-P):NEXT
21060 RETURN
```

Dieses Modul führt dieselbe Funktion aus, wie das Formatiermodul im letzten Programm.

MODUL 6.2.9

```
17000 REM#*****************************
17010 REM AENDERN/LOESCHEN
17020 REM******************************
17030 FOR I=0 TO C(CD)-1
17040 PRINT "██[ROT][REVERS EIN]█████████
 AENDERN ODER LOESCHEN[REVERS AUS]"
17050 IF LEFT$(A$(CD,I),1)<>"$" THEN PRI
NT "██";MID$(A$(CD,I),2);
17060 IF LEFT$(A$(CD,I),1)="*" THEN LET
HH$=MID$(A$(CD,I),2):PRINT
17070 IF LEFT$(A$(CD,I),1)="$" THEN PRIN
T "██";HH$:PRINT"█";MID$(A$(CD,I),2);
17080 IF A(CD,I)=0 THEN 17100
17090 PRINT"██████████████████";:MM=A(CD,I
):GOSUB 21000:PRINT M$
17100 PRINT "██████████[ROT]FUNKTIONSTAST
EN-BEFEHLE:"
17110 PRINT "█[GRUEN]  F1 - NAECHSTE ZAH
LUNG"
17120 PRINT "█  F3 - BETRAG AENDERN"
17130 PRINT "█  F5 - ZURUECK ZUM MENUE"
17140 PRINT "█  F8 - ZAHLUNG LOESCHEN"
17150 PRINT "██[PURPUR]BEFEHL:?"
17160 GET Q$:IF Q$="" THEN 17160
17170 IF Q$=CHR$(140) THEN GOSUB 18000:R
ETURN
17180 IF Q$=CHR$(135) THEN RETURN
17190 IF Q$<>CHR$(134) OR LEFT$(A$(CD,I)
,1)="*" THEN GOTO 17240
```

```
17200 INPUT "[SCHWARZ]ZU ADDIERENDER B
ETRAG:";Q
17210 INPUT "[ROT]EINGABE RICHTIG (J/N)
:";R$
17220 IF R$<>"N" THEN A(CD,I)=A(CD,I)+Q:
GOTO 17040
17230 PRINT "";:FOR L=1 TO 3:PRINT C
$;:NEXT:PRINT "";:GOTO 17200
17240 NEXT I:RETURN
```

Mit diesem Modul kann der Benutzer durch die Zahlungen auf der angegebenen Seite der Konten gehen und einzelne Zahlungen ändern oder löschen.

Kommentar

Zeilen 17050-17090: Diese Bedingungen befassen sich mit der Formatierung der verschiedenen Zahlungstypen. Handelt es sich bei die Zahlung nicht um einen Untertitel, so wird der Name der Zahlung ausgedruckt, wobei der Name auch in HH$ gespeichert wird, wenn es sich um einen Haupttitel handelt. Handelt es sich bei der Zahlung um einen Untertitel, so wird der vorher gespeicherte Haupttitel vor dem Untertitel ausgedruckt, um die Gruppe anzugeben, in die der Titel fällt. Handelt es sich schließlich bei der Zahlung um eine einzelne Zahlung oder einen Untertitel, so wird der damit verknüpfte Betrag mit Hilfe des vorhergehenden Moduls formatiert, bevor er ausgedruckt wird.

Zeilen 17190-17230: Wird die Taste f3 für eine einzelne Zahlung oder einen Untertitel betätigt, so hat der Benutzer die Möglichkeit, den mit einer Zahlung verknüpften Betrag zu ändern, indem er eine positive oder negative Zahl eingibt. Hier wird auf die Benutzung von C$ verwiesen, um die Eingabeaufforderungen im Falle eines Fehlers zu löschen.

MODUL 6.2.10

```
18000 REM#*****************************
18010 REM LOESCHEN
18020 REM******************************
18030 PL=I:IF LEFT$(A$(CD,PL),1)<>"*" TH
EN GR=1:GOTO 18060
18040 GR=0
18050 GR=GR+1:IF LEFT$(A$(CD,PL+GR),1)="
*" THEN GOTO 18050
```

```
18060 FOR K=PL TO C(CD)-GR-1:A(CD,K)=A(C
D,K+GR):A$(CD,K)=A$(CD,K+GR):NEXT
18070 C(CD)=C(CD)-GR:I=I-GR+1:RETURN
```

Dieses Modul nimmt die Löschungen vor, die in dem vorhergehenden Modul angegeben wurden.

Kommentar

Zeile 18030: PL wird gleich dem Wert der Schleifenvariablen in dem vorhergehenden Modul gesetzt. Bei Untertiteln und einzelnen Zahlungen wird die Variable GR, mit der die Anzahl von zu löschenden Zahlungen angegeben wird, gleich 1 gesetzt.

Zeile 18040: Bei Haupttiteln wird die Variable GR erhöht, um den Haupttitel selbst und sämtliche Untertitel zu berücksichtigen, da diese zusammen mit dem Haupttitel gelöscht werden müssen.

Zeile 18060: Mit GR wird festgelegt, wie viele Zahlungen in der Datei überschrieben werden, und um wieviel der Wert auf der jeweiligen Seite des Feldes C gekürzt werden muß.

Testen der Moduln 6.2.8 - 6.2.10

Nun sollte der Benutzer in der Lage sein, Daten einzugeben und durch diese Daten zu gehen, wobei die zugehörigen Beträge geändert oder einzelne Zahlungen gelöscht werden können.

MODUL 6.2.11

```
19000 REM#*****************************
19010 REM RECHNUNG AUSGEBEN
19020 REM******************************
19030 LET PA$="HABEN":IF CD=1 THEN PA$="
SOLL"
19040 TT=0:SS=0:PRINT "[illegible][REVER
S EIN][ROT]";PA$;""
19050 FOR I=0 TO C(CD)-1:TT=TT+A(CD,I)
19060 PRINT "[SCHWARZ]";:IF I/2=INT(I/2)
 THEN PRINT "[GRUEN]";
19070 IF LEFT$(A$(CD,I),1)="*" THEN PRIN
T
```

```
19080 IF LEFT$(A$(CD,I),1)="$" THEN PRIN
T "██";
19090 PRINT MID$(A$(CD,I),2)
19100 IF LEFT$(A$(CD,I),1)="*" THEN 1915
0
19110 PRINT "█████████████████████";
19120 IF LEFT$(A$(CD,I),1)="%" THEN PRIN
T"███████████";
19130 MM=A(CD,I):GOSUB 21000:PRINT M$
19140 IF LEFT$(A$(CD,I),1)="$" THEN SS=S
S+A(CD,I)
19150 IF SS=0 OR LEFT$(A$(CD,I+1),1)="$"
 THEN 19180
19160 PRINT "████████████████████-------██
    ";
19170 MM=SS:GOSUB21000:PRINT M$:SS=0
19180 GET GG$:IF GG$="" THEN 19180
19190 NEXT I:PRINT "███████████████████████
         -------"
19200 PRINT"[REVERS EIN]TOTAL[REVERS AUS
]:                          ";
19210 MM=TT:GOSUB 21000:PRINT M$
19220 PRINT "█[ROT]TASTE DRUECKEN"
19230 GET GG$:IF GG$="" THEN 19230
19240 RETURN
19250 STOP
```

Dieses Modul läuft parallel zu dem Modul für das Ausdrucken von Kontoauszügen im letzten Programm.

Kommentar

Zeile 19060: Um zu verdeutlichen, welche Summe zu welchem Betrag gehört, werden die Zahlungen und die zugehörigen Beträge abwechselnd in schwarz und grün ausgedruckt.

Zeile 19070: Vor einem Haupttitel steht eine Leerzeile.

Zeile 19080: Untertitel werden um zwei Stellen eingerückt.

Zeilen 19100-19140: Für Untertitel und einzelne Zahlungen werden der Zahlungs-

name und der zugehörige Betrag ausgedruckt. Beträge von Untertiteln werden in einer separaten Spalte ausgedruckt. Die Gesamtsumme von Untertiteln unter einem Haupttitel wird in SS kumuliert.

Zeilen 19150-19170: Am Ende einer Gruppe mit Untertiteln wird die Gesamtsumme für die Gruppe ausgedruckt.

Zeile 19180: Erneut werden die Zahlungen nacheinander ausgedruckt, wobei die nächste Zahlung erst gedruckt wird, nachdem eine beliebige Taste betätigt wurde.

Zeilen 19200-19210: Die Gesamtsumme für die jeweilige Seite der Konten wird ausgedruckt.

Testen von Modul 6.2.11

Nachdem einige Daten eingegeben wurden, sollte jede Seite der Konten angezeigt werden können. Hier wird darauf hingewiesen, daß jeweils nur eine Seite gleichzeitig angezeigt werden kann.

Zusammenfassung

Jetzt sollten Sie schon mit den Techniken vertraut sein, die beim Hinzufügen und Löschen von Zahlungen benutzt werden, ohne die Gesamtreihenfolge der Datei zu stören. Außerdem sollten auch Sie schon Freude daran haben, selbst einfache Zahlen sauber formatiert auf dem Bildschirm anzuzeigen. Es lohnt sich durchaus, einige der hier benutzten Methoden zu überprüfen, bevor fortgefahren wird. Im nächsten Programm werden nämlich wesentlich komplexere Daten als bisher verarbeitet und angezeigt.

Ein Schritt weiter

1. Eine weitere nützliche Funktion bestünde in der Möglichkeit, den Saldo zwischen den beiden Seiten des Kontos auszudrucken, wenn eine der beiden Seiten angezeigt wird.
2. Wie schon im vorhergehenden Programm werden Sie das aktuelle Benutzer-Suchmodul ändern wollen, wenn eine große Anzahl von Zahlungen gespeichert werden soll. Mit diesem Modul können die Zahlungen nämlich nur eine nach der anderen überprüft werden. Dennoch muß hier vorsichtig vorgegangen werden, da das Modul Haupttitel entdecken muß, während es durch die Datei geht, insbesondere, wenn diese gelöscht werden müssen. Wird einfach durch die Datei gesprungen, ohne diesen Punkt zu berücksichtigen, so könnte dies veheerende Folgen haben.

6.3 BUDGET

Nun wenden wir unsere Aufmerksamkeit dem komplexesten und schwierigsten Programm in diesem Buch zu. Budget ist eine leistungsfähige und flexible Finanzhilfe, mit der der Benutzer die Finanzen über eine Dauer von zwölf Monaten planen und die Folgen von 'Was-ist-wenn' Entscheidungen über Einnahmen und Ausgaben prüfen kann. Wird dieses Programm richtig eingesetzt, so kann es einen überraschenden Einblick in die Finanzen einer Familie über das kommende Jahr gewähren, ganz abgesehen davon, daß einige der Probleme der Arbeit mit großen Mengen numerischer Daten verdeutlicht werden. Die vom Programm benutzten Felder speichern etwa 800 verschiedene numerische Werte.

Budget: Variablentabelle

BA(1,11)	Kassenbestand jedes Monats.
BD(1,11)	Saldo der geplanten Zahlungen im Vergleich zu den tatsächlichen Zahlungen.
C1(1,11)	Haupteinkommen.
C2(1,11)	Nebeneinkommen.
CU	Temporäre Variable, die bei der Berechnung des kumulativen Überschusses/Defizits benutzt wird.
FO$	Cursor-Kontrollfolge, die bei der Formatierung der Tabelle benutzt wird.
H	Indikator, mit dem angegeben wird, welche Seite der Felder adressiert werden soll.
I1	Variable, mit der die richtige Handhabung von 12-Monats-Zeiträumen gewährleistet wird, die über das Ende des Kalenderjahres hinausgehen.
M1	Temporäre Variable für den Monat, mit dem die Tabellenanzeige begonnen wird.
M2	Temporäre Variable, in der die Änderung des laufenden Monats aufgezeichnet wird.
MM	Nummer des laufenden Monats.
MO(1,29)	Durchschnittliche monatliche Zahlung für jeden Zahlungstitel.
MO$(11)	Monatsnamen.
MY	Temporäre Variable, die bei der Formatierung der Geldbeträge benutzt wird.
MY$	Zeichenfolge, in der der formatierte Geldbetrag gespeichert wird.
N(1)	Anzahl von Zahlungen auf beiden Seiten der Felder.
PA(1,29,11)	Beträge, die mit den Zahlungstiteln verknüpft sind.
PA$(1,29)	Namen der Zahlungstitel.
PP	Temporäre Variable, mit der die Position der Zahlung angegeben wird, die in dem Feld geändert werden muß.
PT(1,11)	Monatliche Gesamtausgaben.

R$	Separator der Datendatei.
T(1)	Temporäre Variable, mit der der Gesamtbetrag berechnet wird, der in der durchschnittlichen Budgetzuweisung beiseite gelegt wird.
TT	Temporäre Variable, die zur Berechnung der Gesamtzahlen für Zahlungen benutzt wird, die in der durchschnittlichen Budgetberechnung enthalten sind.
Y	Nummer des Monats bei Jahresende.

MODUL 6.3.1

```
21000 REM#******************************
21010 REM LADEN/SPEICHERN
21020 REM******************************
21030 PRINT "[ROT]KASSETTE EINLEGEN, DAN
N [REVERS EIN]RETURN[REVERS AUS]--"
21040 INPUT "MOTOR HAELT AUTOMATISCH AN:
";Q$:POKE 192,7:POKE 1,39
21050 PRINT "■[BLAU]MOEGLICHE BEFEHLE:"
21060 PRINT "[GRUEN] 1)DATEN SPEICHERN":
PRINT " 2)DATEN LADEN"
21070 INPUT "[SCHWARZ]BEFEHL:";Q:ON Q GO
TO 21080,21130:RETURN
21080 POKE 1,7:FOR I=1 TO 2000:NEXT
21090 OPEN 1,1,1,"BUDGET":PRINT#1,MM,R$,
Y:FOR H=0 TO 1:PRINT#1,N(H)
21100 FOR I=0 TO 11:PRINT#1,C1(H,I),R$,C
2(H,I):NEXT I
21110 FOR I=0 TO N(H)-1:IF PA$(H,I)="" T
HEN PA$(H,I)=" "
21120 PRINT#1,PA$(H,I):FOR J=0 TO 11:PRI
NT#1,PA(H,I,J):NEXT J,I,H:CLOSE1:RETURN
21130 OPEN 1,1,0,"BUDGET":INPUT#1,MM,Y:F
OR H=0 TO 1:INPUT#1,N(H)
21140 FOR I=0 TO 11:INPUT#1,C1(H,I),C2(H
,I):NEXT I
21150 FOR I=0 TO N(H)-1
21160 INPUT#1,PA$(H,I):FOR J=0 TO 11:INP
UT#1,PA(H,I,J):NEXT J,I
21170 GOSUB 14000:NEXT H:CLOSE1:RETURN
```

Die Komplexität dieses Datendatei-Moduls sollte Sie von der Notwendigkeit überzeugen, Daten in regelmäßigen Abständen zu speichern, um sich vor Fehlern zu schützen, die bei der Eingabe eines derart komplexen Programmes nicht vermieden werden können.

MODUL 6.3.2

```
11000 REM#*******************************
11010 REM MENUE
11020 REM*******************************
11030 POKE 53281,13:PRINT"[ROT][REVERS EIN]BUDGET"
11040 PRINT "[BLAU]MOEGLICHE FUNKTIONEN:"
11050 PRINT " [GRUEN]1)MONATLICHE ANALYSE AUSGEBEN"
11060 PRINT " 2)AENDERUNGEN"
11070 PRINT " 3)NEUE RECHNUNGEN"
11080 PRINT " 4)RECHNUNGEN LOESCHEN"
11090 PRINT " 5)HYPOTHETISCHE DATEN SETZEN"
11100 PRINT " 6)MONAT EINSTELLEN"
11110 PRINT " 7)LADEN/SPEICHERN"
11120 PRINT " 8)INITIALISIEREN"
11130 PRINT " 9)BEENDEN"
11140 INPUT "[ROT]BEFEHLE:";Z:PRINT ""
;:IF Z<5 THEN 11160
11150 ON Z-4 GOSUB 15000,17000,21000,12000,11190:GOTO 11000
11160 PRINT "[PURPUR]1)WAHRE DATEN":PRINT "2)HYPOTHETISCHE DATEN"
11170 INPUT "[BLAU]EINGABE:";H:IF H<1 OR H>2 THEN 11170
11180 PRINT "":H=H-1:ON Z GOSUB 13000,19000,16000,20000:GOTO 11000
11190 PRINT "[ROT][REVERS EIN]PROGRAMM BEENDET":END
```

Ein Standard-Menümodul, mit dem darüber hinaus definiert werden kann, ob die wahre oder hypothetische Seite der Felder adressiert wird. Der Unterschied zwischen diesen beiden Seiten wird später erläutert.

MODUL 6.3.3

```
12000 REM#******************************
12010 REM INITIALISIEREN
12020 REM*******************************
12030 CLR
12040 DIM PA$(1,29),MO(1,29),PA(1,29,11)
,PT(1,11),BD(1,11),C1(1,11),BA(1,11)
12050 DIM C2(1,11):R$=CHR$(13)
12060 DATA JANUAR,FEBRUAR,MAERZ,APRIL,MA
I,JUNI,JULI,AUGUST,SEPTEMBER
12070 DATA OKTOBER,NOVEMBER,DEZEMBER
12080 DIM MO$(11):RESTORE:FOR I=0 TO 11:
READ MO$(I):NEXT
12090 INPUT "■[SCHWARZ]LADEN SIE VON KAS
SETTE (J/N):";Q$:IF Q$="J" THEN GOTO 110
00
12100 INPUT "■[ROT]NUMMER DES JETZIGEN M
ONATS:";MM:MM=MM-1:Y=MM+11
12110 GOSUB 18000:GOSUB 16000:GOTO11000
```

Initialisierungsmodul.

MODUL 6.3.4

```
18000 REM#******************************
18010 REM EINKOMMEN
18020 REM*******************************
18030 PRINT "■■[BLAU]GEBEN SIE IHR HAUPT
EINKOMMEN FUER DIE   FOLGENDEN MONATE EI
N:"
18040 FOR I=MM TO Y:I1=I:IF I1>11 THEN I
1=I1-12
18050 PRINT MO$(I1);":":INPUT "■■■■■■■■■■
     ";C1(H,I1):NEXT I
18060 PRINT "■■ANDERE ERWARTETE EINKUENF
TE:"
18070 FOR I=MM TO Y:I1=I:IF I1>11 THEN I
1=I1-12
18080 PRINT MO$(I1);":":INPUT "■■■■■■■■■■
     ";C2(H,I1):NEXT I
18090 GOSUB 14000:RETURN
```

In dieses Modul wird das Einkommen unter dem Titel 'Haupteinkommen' und 'Nebeneinkommen' eingegeben.

Kommentar

Zeile 18040: Während die Daten in den Feldern in der Reihenfolge von Januar bis Dezember gespeichert werden, kann der 12-Monat-Zeitraum, den das Programm abdecken kann, in einem beliebigen Kalendermonat beginnen. Dementsprechend wird mit der Variablen I1 gewährleistet, daß die Schleife nach Abschluß des zwölften Monats zu der Adresse des ersten Monats des Kalenderjahres geht.

Zeile 18050: Hier wird darauf hingewiesen, wie die Variable H bestimmt, welche Seite der Felder adressiert wird.

Testen der Moduln 6.3.2 - 6.3.4

Durch Einfügen temporärer RETURN-Befehle bei 14000 und 16000 sollte der Benutzer nun Einkommensdaten für die zwölf Monate ab dem gewählten Anfangsmonat eingeben können. Im Augenblick werden die Einkommenszahlen nur in die wahre Seite der Felder eingegeben - dies wird später noch erläutert.

MODUL 6.3.5

```
16000 REM#******************************
16010 REM EINGABE DER RECHNUNGEN
16020 REM*******************************
16030 PRINT "■■[ROT]■■■■■■■■■■[REVERS EIN
]EINGABE DER RECHNUNGEN:"
16040 PRINT"[GRUEN]SETZEN SIE '*' VOR IH
RE EINGABE, WENN  DIE RECHNUNG NICHT IN
 DIE";
16050 PRINT "[HELLBLAU] ANALYSE":PRINT "
MITEINBEZOGEN WERDEN SOLL."
16055 PRINT "'ZZZ' UM INS MENUE ZURUECKZ
UKEHREN"
16060 INPUT "■[ROT]TITEL DER RECHNUNG:";
Q$:IF Q$="ZZZ" THEN GOSUB 14000:RETURN
16070 N(H)=N(H)+1
16080 IF N(H)=30 THEN N(H)=29:PRINT "KEI
N PLATZ MEHR":FORI=1TO 2000:NEXT:RETURN
16090 PA$(H,N(H)-1)=Q$:PRINT "■[BLAU]AUS
```

```
GABEN FUER ";Q$;":"
16100 FOR I=MM TO Y:I1=I:IF I1>11 THEN I
1=I1-12
16110 PRINT MO$(I1):INPUT "[illegible]"
;PA(H,N(H)-1,I1):NEXT I
16120 GOTO 16000
```

In dieses Modul werden Rechnungstitel und zugehörige Beträge eingegeben.

Kommentar

Zeile 16040: Das Programm verfügt über die Möglichkeit, eine durchschnittliche Monatszahl zu berechnen, die die jährlichen Gesamtzahlungen unter einem beliebigen Zahlungstitel abdeckt. Steht vor dem Zahlungsnamen ein *, so wird die jeweilige Zahlung aus dem Verfahren ausgenommen - d.h. sie wird als Einzelzahlung behandelt.

Zeile 16070: Mit der Variablen H wird die eine oder andere Seite des aus zwei Elementen bestehenden Feldes N erhöht. In ihr wird die Anzahl von Zahlungen festgehalten, die auf jeder Seite des Feldes gespeichert werden.

Zeilen 16090-16110: Nachdem der Zahlungstitel angegeben wurde, wird für jeden der zwölf Monate in dem betreffenden Zeitraum eine Eingabe gefordert.

Testen von Modul 6.3.5

Der Benutzer sollte nun eine Reihe von Rechnungen eingeben und sie auf der Null-Seite des Feldes PA$ und PA wiederfinden - wobei auch hier wieder bei der wahren Seite der Felder geblieben wird. Der temporäre RETURN-Befehl bei 14000 muß für diesen Test beibehalten werden.

MODUL 6.3.6

```
14000 REM#*****************************
14010 REM BUDGET ERNEUERN
14020 REM******************************
14030 T(H)=0
14040 FOR I=0 TO N(H)-1:BU=0:IF LEFT$(PA
$(H,I),1)="*" THEN 14060
14050 FOR J=0 TO 11:BU=BU+PA(H,I,J):NEXT
:MO(H,I)=BU/12:T(H)=T(H)+MO(H,I)
```

```
14060 NEXT I
14070 TT=0:CU=0:FOR I=MM TO Y:I1=I+12*(I
>11):PT(H,I1)=0
14080 FOR J=0 TO N(H)-1:PT(H,I1)=PT(H,I1
)+PA(H,J,I1):NEXT J:TT=TT+PT(H,I1)
14090 FOR J=0 TO N(H)-1:IF LEFT$(PA$(H,J
),1)="*" THEN TT=TT-PA(H,J,I1):NEXT J
14100 BD(H,I1)=T(H)*(I-MM+1)-TT:CU=CU+C1
(H,I1)+C2(H,I1)-PT(H,I1):BA(H,I1)=CU
14110 NEXT I:RETURN
```

Dieses Modul führt sämtliche Berechnungen aus, die für das Erstellen der Zahlentabelle erforderlich sind, die hier aufgebaut werden soll.

Kommentar

Zeilen 14040-14060: Die monatlichen durchschnittlichen Budgetzahlen werden berechnet und in der Matrix MO gespeichert. Die kumulative Gesamtsumme für diese Zahlen wird im Feld T gespeichert. Das Verfahren wird nicht für Zahlungstitel ausgeführt, die mit einem * beginnen.

Zeile 14070: Hier wird auf die Benutzung der Booleschen Bedingung (I < 11) zur Berechnung des Wertes von I1 verwiesen. Ist I kleiner als oder gleich 11, so hat diese Bedingung einen Wert Null und wirkt sich auf den Wert von I1 nicht aus. Ist I größer als 11, so nimmt die Bedingung den Wert minus 1 an und kann zur Rücksetzung von I1 benutzt werden.

Zeile 14080: Die Gesamtsumme sämtlicher in einem bestimmten Monat zu bezahlenden Rechnungen wird in der entsprechenden Zeile des Feldes PT kumuliert. In TT steht die kumulative Gesamtsumme dieser monatlichen Gesamtsummen.

Zeile 14090: Von TT werden nun die Beträge subtrahiert, die mit Zahlungen verknüpft sind, die nicht in den durchschnittlichen Budgetberechnungen aufgenommen werden sollen. TT enthält nun die kumulative Gesamtsumme der Zahlungen, die in der durchschnittlichen Budgetberechnung enthalten sind.

Zeile 141000: Der Saldo der geplanten Zahl im Vergleich zu den tatsächlichen Zahlungen wird nun im Feld BD gespeichert, indem die durchschnittliche monatliche Zahlung mit der Anzahl von Monaten multipliziert wird und die tatsächlichen Zahlungen bei geplanten Posten bis zum entsprechenden Monat subtrahiert wer-

den. Der Saldo der beiden Einkommensarten im Vergleich zu den gesamten tatsächlichen Zahlungen für den Monat wird im Feld BA gespeichert.

Testen von Modul 6.3.6

Dieses Modul kann nur schwierig voll getestet werden, bevor nicht das Modul, mit dem die Tabelle angezeigt wird, eingegeben wurde. Es wird jedoch empfohlen, einige Daten einzugeben, da dadurch das Modul aufgerufen und die Syntax überprüft wird. Sind Sie überzeugt, daß das Modul einwandfrei arbeitet, so wird empfohlen, die eingegebenen Daten auf Band zu speichern.

MODUL 6.3.7

```
22000 REM#******************************
22010 REM UNTERPROGRAMME
22020 REM*******************************
22030 MY=INT(ABS(MY)+10000):MY$=MID$(STR
$(MY),3):IF MY>=20000 THEN MY$="####"
22040 RETURN
```

Eine Formatierroutine, die eine vierstellige Zahl falls erforderlich mit führenden Nullen zurückgibt. Ist die verarbeitete Zahl größer als 9999, so wird sie als "####" angezeigt. Damit wird angegeben, daß die Zahl außerhalb des Bereichs liegt, der von diesem Programm genau angezeigt werden kann. Die Programmberechnungen sind davon nicht betroffen.

MODUL 6.3.8

```
13000 REM#******************************
13010 REM AUSGABE
13020 REM*******************************
13030 PRINT "▀████████████████[ROT][REVERS
 EIN]ANALYSE"
13040 INPUT"MIT WELCHEM MONAT BEGINNEN:"
;M1:IF M1<1 OR M1>11 THEN 13040
13050 M1=M1-1:IF MM-M1-12*(M1>MM-1)<4 TH
EN M1=MM-4-12*(MM<5)
13060 PRINT "▀█[GRUEN] ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
13070 PRINT" [GRUEN]▒[BLAU]MONAT        "
;
```

```
13080 FORJ=M1 TO M1+3:PRINT "[GRUEN]▒[B
LAU]";LEFT$(MO$(J+12*(J>11)),3);
13090 NEXT J:PRINT "[GRUEN]▒ [BLAU]B [GR
UEN]▒"
13100 PRINT "[GRUEN] ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
13110 FOR I=0 TO N(H)-1:IF I<>15 THEN 13
140
13120 INPUT "WEITER MIT [ROT][REVERS EIN
]RETURN[REVERS AUS]:";Q$:PRINT ""
13125 FOR J=1 TO 20
13130 PRINT "
             ":NEXT J:PRINT ""
13140 PRINT " [GRUEN]▒[BLAU]";LEFT$(PA$(
H,I),12):PRINT "████████████████";
13150 FOR J=M1 TO M1+3:PRINT "[GRUEN]▒[R
OT]";:MY=INT(PA(H,I,J+12*(J>11))):GOSUB
22030
13160 PRINT MY$;:NEXT J:PRINT "[GRUEN]▒[
ROT]";
13170 MY=INT(MO(H,I)):GOSUB 22030:PRINT
MY$:NEXT I
13180 PRINT " [GRUEN]▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
13190 INPUT "[SCHWARZ][REVERS EIN]RETURN
[REVERS AUS] FUER AUSGABE DER ANALYSE:";
Q$
13200 PRINT"█████":FOR I=1 TO 20:PRINT "
                                       "
13210 NEXT I:RESTORE:FOR J=1 TO 12:READ
A$:NEXT:PRINT "█████[GRUEN]"
13220 DATA TOTAL,BUDGET,BUDGET BAL.,HAUP
TEINK., NEBENEINK.,GESAMTEINK.
13230 DATA GELDGLEICHGEW,ANHAEUFUNG
13240 FOR I=1 TO 8:READ A$:PRINT " [GRUE
N]▒[BLAU]";A$
13250 PRINT "[GRUEN] ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":NEXT I
13260 FO$="█████████████████"
13270 FOR I=M1 TO M1+3:I1=I+12*(I>11):PR
```

```
INT "▓▓▓▓▓"
13280 PRINT FO$;"[GRUEN]▒";:MY=PT(H,I1):
PRINT "[SCHWARZ]";:IF MY<0 THEN PRINT "[
ROT]";
13290 GOSUB 22030:PRINT MY$:PRINT
13300 PRINT FO$;"[GRUEN]▒";:MY=T(H):PRIN
T "[SCHWARZ]";:IF MY<0 THEN PRINT "[ROT]
";
13310 GOSUB 22030:PRINT MY$:PRINT
13320 PRINT FO$;"[GRUEN]▒";:MY=BD(H,I1):
PRINT "[SCHWARZ]";:IF MY<0 THEN PRINT "[
ROT]";
13330 GOSUB 22030:PRINT MY$:PRINT
13340 PRINT FO$;"[GRUEN]▒";:MY=C1(H,I1):
PRINT "[SCHWARZ]";:IF MY<0 THEN PRINT "[
ROT]";
13350 GOSUB 22030:PRINT MY$:PRINT
13360 PRINT FO$;"[GRUEN]▒";:MY=C2(H,I1):
PRINT "[SCHWARZ]";:IF MY<0 THEN PRINT "[
ROT]";
13370 GOSUB 22030:PRINT MY$:PRINT
13380 PRINT FO$;"[GRUEN]▒";:MY=MY+C1(H,I
1)-10000:PRINT "[SCHWARZ]";:IF MY<0 THEN
 PRINT "[ROT]";
13390 GOSUB 22030:PRINT MY$:PRINT
13400 PRINT FO$;"[GRUEN]▒";:MY=MY-PT(H,I
1)-10000:PRINT "[SCHWARZ]";:IF MY<0 THEN
 PRINT "[ROT]";
13410 GOSUB 22030:PRINT MY$:PRINT
13420 PRINT FO$;"[GRUEN]▒";:MY=BA(H,I1):
PRINT "[SCHWARZ]";:IF MY<0 THEN PRINT "[
ROT]";
13430 GOSUB 22030:PRINT MY$:PRINT
13440 FO$=FO$+"██████":NEXT I
13450 PRINT "[SCHWARZ]MOECHTEN SIE NOCHE
INMAL DIE EINZELWERTE":INPUT "SEHEN (J/N
):";Q$
13460 IF Q$="J" THEN 13060
13470 RETURN
```

Im letzten Programm haben wir festgestellt, daß Anzeigemoduln häufig die komplexesten Moduln eines Programms sind, das eine Tabelle mit Daten darstellen soll. Und dieses Programm ist hier sicherlich keine Ausnahme. Nachdem dies gesagt wurde, muß jedoch darauf hingewiesen werden, daß dieses Modul unter seiner oberflächlichen Komplexität ein relativ einfaches Modul ist, das schon berechnete Zahlen aufnimmt und auf dem Bildschirm ausgibt. Es sieht nur wegen seiner großen Anzahl von anzuzeigenden Zahlen so komplex aus.

Kommentar

Zeilen 13040-13050: In dieser Tabelle werden die Zahlen für vier Monate ab dem vom Benutzer angegebenen Monat angezeigt. Wird jedoch über das Ende des laufenden 12-Monat-Zeitraums hinausgegangen, so wird die Tabelle sinnlos. Wird also eine kleinere Zahl als vier Monate ab dem Ende des Zeitraums eingegeben, so wird der Anfangsmonat zurückgesetzt.

Zeilen 13060-13100: Der Titel der Tabelle wird ausgedruckt. Er besteht aus den drei ersten Buchstaben der entsprechenden Monate und einem Titel für die Spalte 'durchschnittliches Budget'.

Zeilen 13100-13170: Die Zahlungsnamen werden aus dem Feld PA$ genommen und in der linken Spalte ausgedruckt. Im Anschluß an den Namen werden die Zahlen für die vier Monate und die durchschnittliche Budgetzahl für die Zahlungen, durch Grafikzeichen getrennt, in die Spalten auf dem Bildschirm gedruckt, wobei die Beträge mit dem vorhergehenden Modul gegebenenfalls mit führenden Nullen formatiert werden. 15 Zeilen werden ausgedruckt, wobei der Bildschirm gelöscht und weitere 15 Zeilen gedruckt werden, wenn dies nicht ausreicht. Das Programm kann maximal 30 Zahlungstitel verarbeiten.

Zeilen 13210-13240: Die Titel für die im zweiten Teil der Tabelle angegebenen Zahlen werden aus den DATA-Anweisungen gelesen und entlang der linken Seite des Bildschirms ausgedruckt - die Tabellenüberschrift bleibt davon unbetroffen (der Titel der Budget-Spalte ist nun redundant, wird jedoch nicht gelöscht).

Zeilen 13260-13440: Trotz ihrer Länge eine einfache Routine, die mit Hilfe der Zeichenfolge FO$ zur Bestimmung der Spaltenposition die entsprechenden Zahlen für jeden Monat entlang dem Bildschirm neben den entsprechenden Titeln ausdruckt. Am Ende jeder Monats-Spalte werden fünf Cursor-Zeichen nach rechts zu FO$ hinzugefügt. Das Verfahren wird in einer neuen Spalte für den nächsten Monat wiederholt. Hier wird auf die Benutzung der roten und schwarzen Steuerzeichen verwiesen, mit denen angegeben wird, ob eine Zahlung positiv oder negativ ist. Außerdem wird darauf hingewiesen, daß die temporäre Variable MY aus

dem Formatier-Modul in den Zeilen 13380 und 13400 dazu benutzt wird, eine Zahl zu einer vorher ausgedruckten Zahl hinzuzufügen. Hierzu müssen die 10000, die bei dem Formatierverfahren hinzugefügt wurden, zuerst subtrahiert werden.

Testen der Moduln 6.3.7 und 6.3.8

Wurden schon einige Daten gespeichert, so sollten diese nun auf dem Bildschirm angezeigt werden können. Um die Tabelle zu überprüfen (und nicht nur darauf, ob sie richtig angezeigt wird), muß die Bedeutung der verschiedenen Zahlen verstanden werden.

GESAMT	Hier handelt es sich um die Gesamtanzahl sämtlicher in dem Monat vorzunehmenden Zahlungen.
BUDGET	Für jeden Monat gleich, handelt es sich hier um die durchschnittliche Summe, die beiseite gelegt werden muß, um sämtliche nicht ausgeschlossenen Rechnungen in dem 12-Monat-Zeitraum bezahlen zu können. Ein durchschnittliches Budget deckt nicht unbedingt sämtliche Zahlungen bis zu einem bestimmten Monat ab (z.B. wenn sämtliche Zahlungen im ersten Monat geleistet wurden). Mit dieser Zahl wird angegeben, ob der im Durchschnittsbudget berücksichtigte Betrag vor oder hinter den tatsächlichen Zahlungen liegt, für die diese Zahl gilt. Am Ende des 12-Monat-Zeitraums, ist sie gleich Null.
HAUPTEINKOMMEN/NEBENEINKOMMEN/GESAMTEINKOMMEN	Diese Begriffe erläutern sich von selbst.
GELDGLEICHGEWICHT	Die Differenz zwischen dem Einkommen und den abgebuchten Beträgen für den jeweiligen Monat.
ANHÄUFUNG	Die Differenz zwischen dem Gesamteinkommen und den Gesamtzahlungen seit Beginn des 12-Monat-Zeitraums.

Hier wird es zu kleinen Diskrepanzen kommen, da nur Ganzzahlen angezeigt werden, während bei der tatsächlichen Berechnung mit vollen Zahlen gearbeitet wird. So wird das monatliche Budget für eine Zahlung von 47 DM als 3 DM angezeigt. Dies beeinflußt jedoch die richtige Berechnung des gesamten Monatsbudgets nicht.

MODUL 6.3.9

```
19000 REM#*****************************
19010 REM AENDERUNGEN
19020 REM******************************
19030 PRINT "■■■■■■■■■■■■■■■■■■■[ROT][REV
ERS EIN]AENDERUNGEN"
19040 PRINT "■[BLAU]MOEGLICHE BEFEHLE:"
19050 PRINT " [GRUEN]1)RECHNUNGSDATEN AE
NDERN"
19060 PRINT " 2)HAUPTEINKOMMEN AENDERN"
19070 PRINT " 3)ZUSATZEINKOMMEN AENDERN"

19080 INPUT "■[SCHWARZ]BEFEHL:";QQ:ON QQ
 GOSUB 19100,19190,19190
19090 GOSUB 14000:RETURN
19100 INPUT "■[ROT]NAME DER ZU AENDERNDE
N RECHNUNG:";Q$
19110 FOR I=0 TO N(H)-1:IF Q$<>PA$(H,I)
THEN 19130
19120 PP=I:GOTO 19140
19130 NEXT I:PRINT "■RECHNUNG NICHT GEFU
NDEN":FOR I=1 TO 2000:NEXT I:RETURN
19140 PRINT "■■";PA$(H,PP):PRINT "■[BLAU
]NEUEN WERT EINGEBEN ODER '*' UM INS"
19145 PRINT "MENUE ZURUECKZUKEHREN:■"
19150 FOR I=MM TO Y:I1=I+12*(I>11)
19160 PRINT MO$(I1):PRINT "■■■■■■■■■■■■■
:";PA(H,PP,I1);:INPUT Q$
19170 IF Q$<>"*" THEN PA(H,PP,I1)=VAL(Q$
)
19180 NEXT I:RETURN
19190 IF QQ=2 THEN PRINT "■[ROT]■HAUPTEI
NKOMMEN:";
19200 IF QQ=3 THEN PRINT "■[ROT]■ZUSAETZ
LICHES EINKOMMEN:";
19205 PRINT " ('*' LAESST DEN WERT UNGEA
ENDERT)"
19210 FOR I=MM TO Y:I1=I+12*(I>11)
19220 PRINT MO$(I1):PRINT "■■■■■■■■■■■■■
```

```
:";
19230 IF QQ=2 THEN PRINT C1(H,I1);
19240 IF QQ=3 THEN PRINT C2(H,I1);
19250 INPUT Q$:IF Q$<>"*" AND QQ=2 THEN
C1(H,I1)=VAL(Q$)
19260 IF Q$<>"*" AND QQ=3 THEN C2(H,I1)=
VAL(Q$)
19270 NEXT I:RETURN
```

Ist eine Änderung an einem schon eingegebenen Posten erforderlich, so wird dieses Modul benutzt. Mit ihm kann der Benutzer angeben, ob es sich bei dem zu ändernden Posten um einen Zahlungstitel, ein Haupt- oder Nebeneinkommen handelt. Die entsprechenden Zahlen werden dann angezeigt und der Benutzer kann jede Zahl durch Eingabe eines * oder durch Eingabe eines neuen Wertes bestätigen.

MODUL 6.3.10

```
20000 REM#*****************************
20010 REM RECHNUNG LOESCHEN
20020 REM******************************
20030 INPUT "■[BLAU]NAME DER ZU LOESCHEN
DEN RECHNUNG:";Q$
20050 FOR I=0 TO N(H)-1:IF Q$=PA$(H,I) T
HEN 20080
20070 NEXT I:PRINT "■[ROT]RECHNUNG NICHT
 GEFUNDEN":FOR J=1 TO 2000:NEXT J:RETURN

20080 N(H)=N(H)-1:FOR J=I TO N(H)-1:PA$(
H,J)=PA$(H,J+1)
20090 FOR K=0 TO 11:PA(H,J,K)=PA(H,J+1,K
):NEXT K,J:GOSUB 14000:RETURN
```

Mit diesem Modul können Budgetüberschriften gelöscht werden.

MODUL 6.3.11

```
17000 REM#*****************************
17010 REM MONATSWERTE EINSTELLEN
17020 REM******************************
17030 PRINT "■■■■■[ROT][REVERS EIN]WERTE
```

```
 FUER BEST. MONAT EINSTELLEN"
17040 PRINT "[BLAU]GEBEN SIE DIE NUMMER
 DES MOMENTANEN":INPUT"MONATS EIN:";M2
17045 IF M2<0 OR M2>12 THEN17040
17050 M2=M2-1:IF M2=MM THEN RETURN
17060 IF M2<MM THEN M2=M2+12
17070 FOR I=MM TO M2-1:I1=I+12*(I>11)
17080 PRINT "[ROT][REVERS EIN]WERTE
 FUER BEST. MONAT EINSTELLEN"
17090 PRINT "[BLAU]GEBEN SIE DIE WERTE
FUER DEN NAECHSTEN":PRINT MO$(I1);" EIN:
"
17100 FOR J=0 TO N(0)-1:PRINT PA$(0,J);"
 (";PA(0,J,I1);"):";:INPUT PA(0,J,I1)
17110 NEXT J
17120 INPUT "[ROT]HAUPTEINKOMMEN:";C1(0
,I1)
17130 INPUT "[GRUEN]ZUSAETZLICHE EINKOM
MEN:";C2(0,I1):NEXT I
17140 MM=M2+12*(M2>11):Y=MM+11:H=0:GOSUB
 14000:GOSUB 15000:RETURN
```

Dieses Modul ermöglicht Änderungen des Monats. Gibt der Benutzer an, daß der laufende Monat geändert wurde, so werden neue Zahlen für jeden Zahlungstitel und für die Einkommenstypen für jeden der vergangenen Monate angefordert. Sie werden an das Ende des 12-Monat-Zeitraums angefügt. Begann der alte Zeitraum mit dem Monat Mai und beginnt der neue Zeitraum mit Juli, so wird der Benutzer also aufgefordert, nur Zahlen für Mai und Juni einzugeben, da diese nun zu den beiden letzten Monaten des 12-Monat-Zeitraums werden.

Testen von Modul 6.3.11

Nun sollte der Benutzer die Zahlen für die Zahlungstitel oder das Einkommen ändern, die Zahlungstitel löschen und den Zeitraum ändern können, den das Programm abdeckt. Um das letzte Modul zu testen, muß bei Zeile 15000 ein temporärer RETURN-Befehl eingefügt werden.

MODUL 6.3.12

```
15000 REM#******************************
15010 REM HYPOTHETISCHE WERTE SETZEN
15020 REM******************************
15030 T(1)=T(0)
15040 FOR I=0 TO N(0)-1:PA$(1,I)=PA$(0,I
):MO(1,I)=MO(0,I)
15050 FOR J=0 TO 11:PA(1,I,J)=PA(0,I,J):
NEXT J,I
15060 FOR J=0 TO 11:PT(1,J)=PT(0,J):BD(1
,J)=BD(0,J):C1(1,J)=C1(0,J)
15070 BA(1,J)=BA(0,J):NEXT J:N(1)=N(0):R
ETURN
```

Dieses einfache Modul ist eines der wichtigsten Moduln in dem Programm. Es kopiert die in die wahre Seite der Felder eingegebenen Daten in die hypothetische Seite. Einer der wichtigsten Punkte dieses Programms besteht darin, daß der Benutzer Daten in die hypothetische Seite der Felder eingeben kann, um die Auswirkungen einer Finanzentscheidung zu testen, ohne daß sich dies auf irgendeine Weise auf die wahren Daten auswirkt.
Sämtliche Operationen des Programms können mit hypothetischen Daten ausgeführt werden. Ist der Benutzer zufrieden, so braucht er nur dieses Modul aufzurufen und die Daten in der hypothetischen Seite werden sofort zu wahren Daten umgesetzt. Dieses Modul wird automatisch aufgerufen, wenn der Monat zurückgesetzt wird. Ansonsten würden die beiden Seiten der Tabellen mit verschiedenen Zeiträumen arbeiten.

Testen von Modul 6.3.12

Nun können die hypothetischen Seiten sämtlicher Funktionen getestet werden, indem einfach hypothetische Daten angegeben werden, wenn die Funktionen aufgerufen werden. Zahlungen werden auf beiden Seiten addiert und subtrahiert. Danach wird anhand der Tabellenanzeige geprüft, ob keine der beiden Seiten die andere beeinflußt. Anschließend werden mit diesem Modul die wahren Daten in die hypothetische Seite kopiert. Hier wird darauf hingewiesen, daß die hypothetische Seite bei der ersten Initialisierung des Programms leer ist.
Arbeiten die Funktionen der hypothetischen Seite einwandfrei, so kann das Programm benutzt werden.

Zusammenfassung

Dieses lange Programm ist sehr leistungsfähig, wenn es richtig eingesetzt wird. Allerdings ist einige Übung erforderlich, um es maximal nutzen zu können. Wird es ernst genommen, so kann es einige überraschende Informationen über den Status der Finanzen im Laufe des ganzen Jahres vermitteln. So gibt es an, wann die Geldmittel etwas knapp werden und wann etwas mehr Geld zur Verfügung steht, wie die Zahlungen neu angeordnet werden können, um etwas mehr Geld zu Weihnachten oder im Urlaub zur Verfügung zu haben, wie sich eine neue Verpflichtung oder ein höheres Einkommen auswirkt.
Hier darf jedoch nicht vergessen werden, daß der Commodore 64 mit diesem Handbuch zum Einsatz durch den Benutzer vorbereitet werden soll. Konnte dieses Programm problemlos ausgetestet werden, so gibt es keinen Grund, warum es nicht an andere Anwendungen angepaßt werden sollte, die eine flexiblere Eingabe und Verarbeitung von Daten erfordern, zusammen mit einer klaren Präsentation in Tabellenform und der Möglichkeit, zwei Datengruppen gleichzeitig auszuführen. Das Programm kann als Grundlage für den vertrauensvollen Einsatz des Commodore 64 betrachtet werden.

Ein Schritt weiter

1. Dieses Programm könnte noch nützlicher sein, bestünde die Möglichkeit, die hypothetischen Felder in wahre Felder zu kopieren, sobald sich der Benutzer entscheidet, mit geschätzten Daten weiterzuarbeiten. Dies setzt nur eine kleine Änderung an einem Modul voraus.
2. An der Länge des Programms könnten Einsparungen vorgenommen werden, indem die Anzahl von Feldern reduziert wird. Dies könnte geschehen, indem dieselbe Datenmenge in weniger, jedoch komplexere Feldern gesetzt wird. Dann könnten die Daten unter Umständen mit einer kleinen Anzahl von Schleifen ausgedruckt werden.
3. Soll nur ein einziger Wert für eine Zahlung oder für das Einkommen geändert werden, so müssen alle zwölf Zahlungen durchgegangen werden. Hier könnte versucht werden, eine Funktion hinzuzufügen, mit der in die Mitte des Zeitraums gesprungen und aus der Serie herausgegangen werden kann, sobald die gewünschte Änderung vorgenommen wurde.

KAPITEL 7

MUSIC

Eine der erfreulichsten Seiten des Commodore 64 ist die Art und Weise, mit der Qualität und Vielfalt der Tonmöglichkeiten eine neue Welt für die Benutzer von Homecomputern eröffnen. In nicht allzuferner Zukunft werden fraglos ganze Bücher über die Benutzung des Sound Interface Device (SID) Chips des C64 geschrieben werden.

Die große Komplexität der Möglichkeiten des SID-Chips bedeutet, daß kein einzelnes Programm ihnen voll gerecht werden kann und daß ein Kapitel eines allgemin gehaltenten Buches nur eine Einführung in die nahezu grenzenlosen Kombinationen der zur Verfügung stehenden Töne darstellen kann. Nachdem dies gesagt wurde, wird jedoch hier ein Programm vorgestellt, das eine feste Grundlage für künftige Experimente und Tonschöpfungen darstellt. Dieses Programm soll dem Benutzer nicht nur ermöglichen, Töne einzugeben und zu spielen (was der Fall ist), sondern soll ihm jeden Teil des SID direkt zugänglich machen. Ein Großteil der Möglichkeiten des SID werden einfache durch Benutzung dieses Programms eingesetzt.

Als erstes muß daran gedacht werden, daß eine normale Musiknote nicht nur eine Vibration mit einer bestimmten Frequenz darstellt, sondern in Wirklichkeit eine Kombination verschiedener hoher und niedriger Frequenzen ist. Um eine Note zu erzeugen, müssen demzufolge zwei separate Frequenzen, eine hohe und eine niedrige, eingegeben werden. Jede der drei Stimmen des SID sieht diese beiden Eingaben für jede gespielte Note vor. Das Programm muß in der Lage sein, Noten in einer dem Benutzer verständlichen Form zu akzeptieren und diese Noten dann in eine von dem SID benutzbare Form zu übersetzen.

Zweitens variiert die Intensität einer bestimmten Note auf komplexe Weise, während die Note gespielt wird.

a) Die erste Phase der Note wird als ANSCHLAG bezeichnet. Hier handelt es sich um die Geschwindigkeit, mit der der Ton von Nichts zur Spitze ansteigt. Je kürzer die Dauer des Anschlags, desto schärfer der Ton.

b) Die zweite Phase wird als ABSCHWELLUNG bezeichnet. Während dieser Phase fällt die Note von ihrer ursprünglichen Spitze ab.

c) Nach diesem ersten Abfallen geht die Note in die HALTE-Phase, mit der die Länge des Haupttons bestimmt wird.

d) Als letztes geht die Note in die Phase des AUSKLINGENS, die wie die ANSCHLAG-Phase scharf oder gemächlich sein kann.

Verschiedene Musikinstrumente verfügen über verschiedene Tonqualitäten, un-

abhängig von den gespielten Noten und der Form dieser Noten. Diese Unterschiede hängen von der Wellenform des Tons ab, den das Instrument erzeugt.
Bei dem SID kann jede der drei Stimmen eine von drei musikalischen Wellenformen erzeugen und eine weitere weiße Rausch-Wellenform, die beim Erzeugen von Rauscheffekten von besonderem Nutzen ist.
Von den drei musikalischen Wellenformen kann eine, die Rechteckfrequenz, selbst einen beträchlichen Grad der Variation erzeugen, indem die Länge der Impulse geändert wird, aus denen die Frequenz besteht.
Nachdem schließlich die gewünschte Frequenz, der Ton und die Form der Note erzeugt wurden, können die Noten mit dem SID-Chip gefiltert werden. Dies bedeutet, daß verschiedene Frequenzen innerhalb der Note in der Lautstärke vermindert werden können, während andere unberührt bleiben.

Music: Variablentabelle

FI%(3):	Filtereigenschaften für die drei Stimmen.
HF%(2,1000):	Hochfrequenz-Werte für jede Note in der zu spielenden Melodie.
IN:	Initialisierungs-Zeiger.
LF%(2,1000):	Niederfrequenz-Werte für jede Note in der zu spielenden Melodie.
RS:	Separator für Datendateien.
NL:	Länge der Note.
NO%(1,95):	Hoch- und Nieder-Frequenzwerte für die 96 zur Verfügung stehenden Noten.
NT:	Notenwert aus Anhang M des Bedienungshandbuchs.
VO%(2,6):	Vom Benutzer definierte Werte, die mit POKE in den SID-Chip gesetzt werden müssen, um die Toneigen schaften der drei Stimmen festzulegen.
VS:	Anfangsadresse einer Stimme in dem SID.
WF%(2,1000):	Wellenform-Werte für jede zu spielende Note.
WW:	Wellenform-Wert für jede einzelne Note.

MODUL 7.1.1

```
11000 REM#*****************************
11010 REM MENUE
11020 REM******************************
11030 POKE 53281,15:PRINT "
  [ROT][REVERS EIN]MUSIC"
11040 PRINT "[BLAU]MOEGLICHE BEFEHLE:"
11050 PRINT "[GRUEN]1)STIMMEN EINSTELLE
N"
11060 PRINT"2)MELODIE ABSPIELEN"
11070 PRINT"3)NOTEN BERECHNEN"
```

```
11080 PRINT"4)LADEN/SPEICHERN"
11090 PRINT"5)INITIALISIEREN"
11100 PRINT "6)BEENDEN
11110 INPUT "[ROT]BEFEHL:";Z:PRINT "";
11120 IF IN=0 AND (Z<5) THEN PRINT "[SCH
WARZ]NOCH NICHT INITIALISIERT!"
11125 FOR I=1 TO 2000:NEXT:GOTO 11000
11130 ON Z GOSUB 15000,14000,13000,17000
,12000,11140:GOTO 11000
11140 PRINT "
[ROT][REVERS EIN]PROGRAMM BEENDET":END
```

Ein Standard-Menümodul

MODUL 7.1.2

```
18000 REM#****************************
18010 REM DATEN FUER NOTEN-TABELLE
18020 REM*****************************
18030 REM NOTEN FREQUENZEN
18040 DATA 268,284,301,318,337,358,379,4
01,425,451,477,506
18050 DATA 536,568,602,637,675,716,758,8
03,851,902,955,1012
18060 DATA 1072,1136,1204,1275,1351,1432
,1517,1607,1703,1804,1911,2025
18070 DATA 2145,2273,2408,2551,2703,2864
,3034,3215,3406,3608,3823,4050
18080 DATA 4291,4557,4817,5103,5407,5728
,6069,6430,6812,7217,7647,8101
18090 DATA 8583,9094,9634,10207,10814,11
457,12139,12860,13625,14435,15294,16203
18100 DATA 17167,18188,19269,20415,21629
,22915,24278,25721,27251,28871,30588
18110 DATA 32407
18120 DATA 34334,36376,38539,40830,43258
,45830,48556,51443,54502,57743,61176
18130 DATA 64814
```

Die Daten in dieser Tabelle stellen einfach eine Abkürzung für die Eingabe der Notenwerte mit hoher und niedriger Frequenz dar. Jede Zahl stellt 256 mal den

Wert der Note mit der hohen Frequenz plus dem Wert der Note mit der niedrigen Frequenz dar.

MODUL 7.1.3

```
12000 REM#******************************
12010 REM INITIALISIEREN
12020 REM*******************************
12030 CLR:DIM NO%(1,95):FOR I=0 TO 95:RE
AD NN:NO%(0,I)=INT(NN/256)
12040 NO%(1,I)=NN-256*INT(NN/256):NEXT
12050 DIM VO%(2,6),FI%(3),LF%(2,1000),HF
%(2,1000),WF%(2,1000)
12070 IN=1:R$=CHR$(13)
12080 GOTO 11000
```

Die Benutzung der hier definierten Hauptvariablen wird in der Variablentabelle erläutert.

Kommentar

Zeilen 12030-12040: Die Hoch- und Niederfrequenz-Werte für jede der 95 Noten werden gelesen und entschlüsselt. Sie können nicht in Form einer einzelnen Zahl im Feld gespeichert werden, da es sich um ein ganzzahliges Feld handelt, die nur Zahlen bis zu einem Wert von 32767 aufnehmen kann. Die Notenwerte werden in NO%(0) für die hohe Frequenz und NO%(1) für die niedrige Frequenz gesetzt.

Testen von Modul 7.1.3

Nachdem dieses Modul aufgerufen wurde, sollten aus der Tabelle Hoch- und Niederfrequenzwerte gelesen werden können, die etwa den Werten in Anhang M des Bedienungshandbuchs entsprechen. Sie werden nicht genau identisch sein, da die hier benutzten Werte aus dem *Programmierhandbuch* (Band 1 der Sachbuchreihe) stammen, in dem eine etwas andere Tabelle steht.

MODUL 7.1.4

```
15000 REM#******************************
15010 REM SETZEN DER STIMMEN
15020 REM*******************************
15030 INPUT "■■[GRUEN]STIMME NUMMER (1-3
):";V:IF V<1 OR V>3 THEN 15030
15040 VS=54272+7*(V-1)
```

```
15050 PRINT "████████████[ROT][REVERS EI
N]STIMME";V
15060 REM*****************************
15070 T1$="FREQUENZ (LOW-BIT:0-255):"
15080 PRINT "[SCHWARZ]";T1$;VO%(V-1,2);:
Q$=""
15090 INPUT Q$:IF Q$<>"" THEN VO%(V-1,2)
=VAL(Q$)
15100 REM*****************************
15110 T1$="FREQUENZ (HIGH-BIT:0-15):"
15120 PRINT T1$;VO%(V-1,3) AND 15;:Q$=""
15130 INPUT Q$:IF Q$<>"" THEN VO%(V-1,3)
=VAL(Q$)
15140 REM*****************************
15150 T1$="RAUSCHGENERATOR (1=AN/0=AUS):
"
15160 PRINT T1$;(VO%(V-1,4) AND 128)/128
;:Q$=""
15170 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 126) OR (VAL(Q$)*129)
15180 REM*****************************
15190 T1$="RECHTECKFREQUENZ (1=AN/0=AUS)
:"
15200 PRINT T1$;(VO%(V-1,4) AND 64)/64;:
Q$=""
15210 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 190) OR (VAL(Q$)*65)
15220 REM*****************************
15230 T1$="SAEGEZAHNFREQUENZ (1=AN/0=AUS
):"
15240 PRINT T1$;(VO%(V-1,4) AND 32)/32;:
Q$=""
15250 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 222) OR VAL(Q$)*33
15260 REM*****************************
15270 T1$="DREIECKFREQUENZ (1=AN/0=AUS):
"
15280 PRINT T1$;(VO%(V-1,4) AND 16)/16;:
Q$=""
15290 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
```

```
=(VO%(V-1,4) AND 238) OR VAL(Q$)*17
15300 REM*****************************
15305 T1$="DIESE STIMME AUSSCHALTEN (1=J
A/0=NEIN):"
15310 PRINT T1$;(VO%(V-1,4) AND 8)/8;:Q$
=""
15320 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 246) OR VAL(Q$)*9
15330 REM*******************************
15340 T1$="RINGMODULATION"+STR$(V)+" MIT
"+STR$(V-1-3*(V=1))+"(1=AN/0=AUS):"
15350 PRINT T1$;(VO%(V-1,4) AND 4)/4;:Q$
=""
15360 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 250) OR VAL(Q$)*5
15370 REM*******************************
15380 T1$="SYNCHRONISATION"+STR$(V)+" MI
T"+STR$(V-1-3*(V=1))+"(1=AN/0=AUS):"
15390 PRINT T1$;(VO%(V-1,4) AND 2)/2;:Q$
=""
15400 INPUT Q$:IF Q$<>"" THEN VO%(V-1,4)
=(VO%(V-1,4) AND 253) OR VAL(Q$)*2
15410 REM*******************************
15420 T1$="STIMMANSCHLAG (0-15):"
15430 PRINT T1$;(VO%(V-1,5) AND 240)/16;
:Q$=""
15440 INPUT Q$:IF Q$<>"" THEN VO%(V-1,5)
=(VO%(V-1,5) AND 15) OR VAL(Q$)*16
15450 REM*******************************
15460 T1$="STIMMABSCHWELLUNG (0-15):"
15470 PRINT T1$;VO%(V-1,5) AND 15;:Q$=""
15480 INPUT Q$:IF Q$<>"" THEN VO%(V-1,5)
=(VO%(V-1,5) AND 240) OR VAL(Q$)
15490 REM*******************************
15500 T1$="HALTEN DER STIMME (0-15):"
15510 PRINT T1$;(VO%(V-1,6) AND 240)/16;
:Q$=""
15520 INPUT Q$:IF Q$<>"" THEN VO%(V-1,6)
=(VO%(V-1,6) AND 15) OR VAL(Q$)*16
15530 REM*******************************
```

```
15540 T1$="AUSKLINGEN (0-15):"
15550 PRINT T1$;VO%(V-1,6) AND 15;:Q$=""
15560 INPUT Q$:IF Q$<>"" THEN VO%(V-1,6)
=(VO%(V-1,6) AND 240) OR VAL(Q$)
15570 REM****************************
15580 T1$="FILTER CUTOFF LOW (0-7):"
15590 PRINT T1$;FI%(0) AND 7;:Q$=""
15600 INPUT Q$:IF Q$<>"" THEN FI%(0)=(FI
%(0) AND 248) OR VAL(Q$)
15610 REM****************************
15620 T1$="FILTER CUTOFF HIGH (0-255):"
15630 PRINT T1$;FI%(1);:Q$=""
15640 INPUT Q$:IF Q$<>"" THEN FI%(1)=VAL
(Q$)
15650 REM****************************
15660 T1$="FILTERRESONANZ (0-15):"
15670 PRINT T1$;(FI%(2) AND 240)/16;:Q$=
""
15680 INPUT Q$:IF Q$<>"" THEN FI%(2)=(FI
%(2) AND 15) OR VAL(Q$)*16
15690 REM****************************
15700 T1$="DIESE STIMME FILTERN (1=JA/0=
NEIN):"
15710 PRINT T1$;(FI%(2) AND 2↑(V-1))/2↑(
V-1);:Q$=""
15720 INPUTQ$:IFQ$<>"" THEN FI%(2)=(FI%(
2) AND (255-2↑(V-1))) OR VAL(Q$)*2↑(V-1)
15730 REM****************************
15740 IF V<> 3 THEN 15780
15750 T1$="CUT-OFF STIMME 3 (1=JA/0=NEIN
):"
15760 PRINT T1$;(FI%(3) AND 128)/128;:Q$
=""
15770 INPUT Q$:IF Q$<>"" THEN FI%(3)=(FI
%(3) AND 127) OR VAL(Q$)*128
15780 REM****************************
15790 T1$="HOCH-PASS FILTER (1=AN/0=AUS)
:"
15800 PRINT T1$;(FI%(3) AND 64)/64;:Q$="
"
```

```
15810 INPUT Q$:IF Q$<>"" THEN FI%(3)=(FI
%(3) AND 191) OR VAL(Q$)*64
15820 REM*******************************
15830 T1$="BAND-PASS FILTER (1=AN/0=AUS)
:"
15840 PRINT T1$;(FI%(3) AND 32)/32;:Q$="
"
15850 INPUT Q$:IF Q$<>"" THEN FI%(3)=(FI
%(3) AND 223) OR VAL(Q$)*32
15860 REM*******************************
15870 T1$="LOW-PASS FILTER (1=AN/0=AUS):
"
15880 PRINT T1$;(FI%(3) AND 16)/16;:Q$="
"
15890 INPUT Q$:IF Q$<>"" THEN FI%(3)=(FI
%(3) AND 239) OR VAL(Q$)*16
15900 REM*******************************
15910 T1$="LAUTSTAERKE (0-15):"
15920 PRINT T1$;FI%(3) AND 15;:Q$=""
15930 INPUT Q$:IF Q$<>"" THEN FI%(3)=FI%
(3) OR VAL(Q$)
15940 FOR I=0 TO 6:IF VO%(V-1,I)>255 THE
N GOTO 15970:NEXT
15950 FOR I=0 TO 3:IF FI%(I)>255 THEN GO
TO 15970:NEXT
15960 RETURN
15970 PRINT"FEHLERHAFTE EINGABE !"
15980 PRINT "BITTE UEBERARBEITEN SIE DIE
S EINGABE     NOCHEINMAL."
15990 FOR I=1 TO 2000:NEXT:GOTO 15000
```

Obwohl dieses Modul sehr lang erscheint, ist es in Wirklichkeit äußerst einfach. Es soll dem Benutzer ermöglichen, sämtliche relevanten Funktionen in dem SID-Chip separat zu adressieren. Die Werte werden dann in die Felder VO% und FI% gespeichert, bis eine Melodie gespielt werden soll.

Kommentar

Zeilen 15030-15040: V ist gleich der Stimm-Nummer, die der Benutzer setzen möchte. Die Adresse 54272 ist der Anfang des SID-Chips, wobei die Hauptteile jeder Stimme sieben Datenbytes belegen, so daß die Anfangsposition der entsprechenden Stimme mit 15040 berechnet wird.

Zeilen 15060-15130: Diese beiden Routinen legen die Schnittlänge fest, wenn der Benutzer die Rechteckfrequenz benutzen möchte. Der aktuelle Wert jeder Routine wird angezeigt und bleibt unverändert, wenn die RETURN-Taste betätigt wird.

Zeilen 15140-15170: Diese Routine legt die zufällige Rauschfrequenz für die Stimme fest. Hier wird darauf hingewiesen, daß wir hier nicht ein gesamtes Byte im Computer adressieren, sondern nur ein Bit (in jedem Byte sind acht Bits oder Ein-/Aus-Schalter vorhanden). Hierzu werden die AND- und OR-Funktionen benutzt. Um anzugeben, ob ein bestimmtes Bit gesetzt ist, wird der Wert in dem Feld ausgedruckt, nachdem er mit AND mit 2 in die Potenz der Nummer des entsprechenden Bits erhoben wurde - die Bits sind von 0 bis 7 in aufsteigender Reihenfolge von rechts nach links numeriert. Ist das Bit gesetzt (eingeschaltet), so wird derselbe Wert zurückgegeben. Ist es nicht gesetzt, so wird der Wert 0 zurückgegeben. Um den zurückgegebenen Wert entweder zu einer Null oder einer Eins zu machen, wird er durch zwei in der Potenz der Bitnummer dividiert. Um den Wert des gewünschten Bits zu ändern, muß der Wert des ganzen Bytes mit AND mit 255-2↑ Bitnummer verknüpft werden. Dadurch wird das gewünschte Bit auf Null gesetzt und alle anderen Bits bleiben unverändert. Das einzelne Bit wird nun mit OR mit der eingegebenen 1 oder 0 verknüpft, so daß sein Wert je nachdem entweder auf 0 oder 1 gesetzt wird.

Zeile 15170: Auch wenn das Bit, das gesetzt werden soll, die Nummer 7 (den Wert 128) aufweist, verknüpfen wir in Wirklichkeit das Byte durch OR mit 129, so daß Bit 7 und Bit 0 gesetzt werden. Dies ist erforderlich, da die Frequenzwerte erst dann einen Ton erzeugen, wenn Bit 0 gesetzt ist.

Zeilen 15180-15290: Diese drei Routinen führen dieselbe Funktion für die Bits 6-4 aus, die drei anderen Frequenzen.

Zeilen 15300-15400: Mit diesen beiden Routinen kann der Benutzer die Ausgabe dieser Stimme mit der Frequenz und der Notenform einer anderen Stimme modulieren, wobei es oft zu überraschenden Ergebnissen kommt. Die andere Stimme braucht nicht unbedingt zu spielen, für sie müssen jedoch Frequenz- und Notenformwerte eingegeben werden. Der Einsatz dieser beiden Funktionen wird nur durch Experimentieren verständlich.

Zeilen 15410-15560: Mit diesen vier Routinen kann ANSCHLAG, ABSCHWELLUNG, HALTEN und AUSKLINGEN gesetzt werden. In diesen Routinen werden keine einzelnen Bits gesetzt, sondern Gruppen von jeweils vier Bits. Wird der Bytewert mit AND mit 240 und danach mit OR mit einem Wert von 0 bis 15 verknüpft, so wirkt sich dies auf die Bits 0 bis 3 aus. Wird das Byte mit AND mit 15 verknüpft und danach mit OR mit einem Wert (0 - 15)*16 verknüpft, so wirkt sich dies auf die

Bits 4 bis 7 aus.

Zeilen 15570-15680: Die Frequenzen, mit denen die Filter des SID arbeiten, können vom Benutzer mit diesen drei Routinen gesetzt werden. Diese Werte gelten dann für sämtliche Stimmen, für die Filter gesetzt sind.

Zeilen 15690-15890: In den restlichen Abschnitten kann der Benutzer die drei zur Verfügung stehenden Filtertypen auf ein oder aus setzen. Der Hochpaßfilter überträgt unveränderte Frequenzen über dem vorher gesetzten Wert. Der Niederpaßfilter führt dieselbe Funktion für die niedrigen Frequenzen aus. Mit dem Bandpaßfilter kann durch ein Band mit Frequenzen im mittleren Bereich gegangen werden. Sind alle drei Filter gesetzt, so wird das Volumen der ganzen Note reduziert. Der Benutzer kann wählen, ob eine bestimmte Stimme gefiltert wird oder nicht.

Zeilen 15730-15770: Bei Stimme 3 ist ein besonderes Bit vorhanden, mit dem die Ausgabe der Stimme abgeschnitten werden kann.

Zeilen 15910-15930: Das Volumen, mit dem Noten gespielt werden, wird für alle Stimmen gleichzeitig gesetzt.

Zeilen 15940-15990: Da bei der Eingabe von Werten bis zu dieser Stelle keine Fehlerprüfungen vorgenommen wurden, wird der Inhalt der Felder überprüft, um festzustellen, ob keine größeren Wert als 255 vorhanden sind, da der Versuch, einen derartigen Wert mit POKE in ein einzelnes Byte zu setzen, zum Programmstopp führen würde.

Testen von Modul 7.1.4

An dieser Stelle kann das Modul nicht voll überprüft werden, da keine Routine vorhanden ist, mit der eine Melodie gespielt werden kann. Allerdings kann eine vernünftige Überprüfung vorgenommen werden, indem die durchgeführten Eingaben sorgfältig festgehalten werden. Mit Hilfe der Auflistung können dann die Werte in den Feldern VO% und FI% ausgedruckt werden, um zu sehen, ob sie mit der Eingabe übereinstimmen. Wird beispielsweise der Höchstwert für jede Eingabeaufforderung eingegeben, während die Stimme auf 1 gesetzt ist, so müßte 255 in VO% (0,2-6) stehen.

MODUL 7.1.5

```
13000 REM#******************************
13010 REM BERECHNUNG
```

```
13020 REM********************************
13030 RESTORE:FOR I=0 TO 95:READ A:NEXT
13040 TL=0:FOR I=0 TO 2:VL=1
13050 READ NT,NL:IF NT=0 THEN 13140
13060 WW=VO%(I,4):IF NT<0 THEN NT=-NT:WW
=1
13065 NT=12*INT(NT/16)+NT-16*INT(NT/16)
13070 IF NL<>1 THEN 13090
13080 HF%(I,VL)=NO%(0,NT):LF%(I,VL)=NO%(
1,NT):WF%(I,VL)=WW:VL=VL+1:GOTO 13050
13090 FOR J=1 TO NL-1:HF%(I,VL)=NO%(0,NT
):LF%(I,VL)=NO%(1,NT):WF%(I,VL)=WW
13100 VL=VL+1:NEXT J
13110 HF%(I,VL)=NO%(0,NT):LF%(I,VL)=NO%(
1,NT):WF%(I,VL)=WW-1:VL=VL+1
13120 IF WF%(I,VL-1)<0 THEN WF%(I,VL-1)=
1
13130 GOTO 13050
13140 IF VL>TL THEN TL=VL
13150 NEXT I
13160 RETURN
```

Dieses Modul nimmt den vom Benutzer in Form von Datenanweisungen angegebenen Ton und kompiliert ihn in eine Form, die von dem SID gespielt werden kann. Dies ist erforderlich, damit das Programm verschiedene Notenlängen berücksichtigen kann. Die tatsächlich vom Programm gespielten Noten weisen alle dieselbe Länge auf, die von einer Zeitschleife festgelegt wird. Längere Noten werden gespielt, indem eine Reihe von Noten mit der angeforderte Länge zusammen gespielt werden, wobei zwischen den einzelnen Teilen der Noten keine Absätze festgestellt werden können. Die Notenlängen können von Stimme zu Stimme variieren, müssen jedoch so beschaffen sein, daß sämtliche benutzten Stimmen koordiniert sind.
Um die Daten für eine Melodie festzulegen, brauchen nur für jede Note ihre Nummer aus Anhang M des Bedienungshandbuchs und ihre Länge angegeben zu werden. Die Einheiten, in denen die Länge aufgezeichnet wird, hängen von der kürzesten Note ab, die gespielt werden soll. Durch Verkürzung der Zeitschleife bei Zeile 14180 können kürzere Noten gespielt werden. Dies bedeutet jedoch, daß längere Noten aus mehreren der kurzen Einheiten erstellt werden müssen. Der Nachteil hierbei besteht darin, daß jede einzelne Einheit einer Note, unabhängig davon, wie kurz oder lang sie von der Zeitschleife gestaltet wird, Platz in jedem der Felder LF%, HF% und WF% belegt, so daß mit Verkürzung der Zeit-

schleife der für die Aufnahme einer Melodie mit einer bestimmten Länge erforderliche Speicherplatz proportional anwächst.

Kommentar

Zeile 13030: Da die Stimmeinstellungen und die Notenwerte während der Entwicklung der Melodie geändert werden, müssen die Daten der Melodie mehrmals gelesen werden. Da die Tabelle mit Notenwerten vor den Melodiedaten stehet, muß der DATEN-Zeiger mit RESTORE zurückgesetzt und die gesamte Notentabelle jedesmal bis zum Anfang der Melodiedaten gelesen werden. Der DATEN-

Zeiger kann nicht auf einen bestimmten gewünschten Punkt in dem Programm gesetzt werden. Er kann nur an den Anfang der Daten zurückgesetzt werden oder weiterhin auf das Datenelement zeigen, das auf das letzte gelesene Element folgt.

Zeile 13040: Diese Schleife gewährleistet, daß die Melodiedaten für jede Stimme kompiliert werden. Mit der Variablen VL wird die Länge der Meldie für jede einzelne Stimme gespeichert.

Zeile 13050: Notenwert und Notenlänge (NL) werden aus den Melodiedaten gelesen. Ist der Notenwert gleich Null, so geht das Programm davon aus, daß die Daten dieser Stimme vollständig sind und geht zur nächsten.

Zeile 13060: Der Wert für die Frequenz der aktuellen Stimme wird aus dem Feld VO% gelesen. Handelt es sich bei dem Notenwert um eine negative Zahl, so wird der Wert der Frequenz um 1 gekürzt, so daß Bit 0 des Frequenzwertes ausgeschaltet wird. Dadurch ergibt sich eine Ruhepause mit einer Länge von NL anstelle einer Note mit einem Ton.

Zeilen 13070-13080: Ist die Notenlänge gleich 1, so werden die hohe Frequenz und die Frequenzdaten in den entsprechenden Feldern gespeichert. Die Stimmlänge wird um 1 erhöht.

Zeilen 13090-13130: Ist die Notenlänge größer als 1, so werden NL-1 aufeinanderfolgende Leerstellen im Feld mit den Frequenz- und Wellenform-Werten gefüllt. Bei der letzten Einheit der Note wird der Wellenform-Wert um 1 gekürzt, so daß die Note natürlich ausklingen kann. Werden für eine Stimme keine anderen Daten als 0,0 eingegeben (d.h. soll diese Stimme nicht benutzt werden), so kann der Frequenz ein Wert von -1 zugewiesen werden. Dadurch stoppt das Programm, falls versucht wird, diesen Wert mit POKE in das Programm zu setzen. Mit Zeile 13120 wird geprüft, ob dies geschehen ist.

Zeile 13140: Die Länge der Melodie für die aktuelle Stimme (VL) wird mit TL verglichen. TL enthält zuerst Null und danach die Länge der längsten Stimme/Melodie. Dadurch wird gewährleistet, daß die Melodie bei Abspielen nicht stoppt, bevor der gesamte Inhalt jeder Stimme/Melodie erschöpft ist.

Testen von Modul 7.1.5

Auch dieses Modul kann erst vollständig getestet werden, wenn die Melodie wirklich gespielt wird. Werden jedoch einige Melodiearten eingegeben, wie in Modul 8 angegeben, wobei mindestens für Stimme 1 eine Wellenform festgelegt wird, so sollten durch Aufruf dieses Moduls die richtigen Hoch- und Niederfrequenzwerte in HF% und LF%, und die Wellenform-Werte in WF% gesetzt werden.

MODUL 7.1.6

```
14000 REM#******************************
14010 REM ABSPIELEN DER MELODIE
14020 REM*******************************
14030 FOR I=54272 TO 54296:POKE I,0:NEXT
14040 FOR I=0 TO 2:VS=54272+7*I
14050 POKE VS+2,VO%(I,2)
14060 POKE VS+3,VO%(I,3)
14070 POKE VS+5,VO%(I,5)
14080 POKE VS+6,VO%(I,6)
14090 NEXT I
14100 POKE 54293,FI%(0)
14110 POKE 54294,FI%(1)
14120 POKE 54295,FI%(2)
14130 POKE 54296,FI%(3)
14140 FOR I=1 TO TL
14150 POKE 54272,LF%(0,I):POKE 54279,LF%
(1,I):POKE 54286,LF%(2,I)
14160 POKE 54273,HF%(0,I):POKE 54280,HF%
(1,I):POKE 54287,HF%(2,I)
14170 POKE 54276,WF%(0,I):POKE 54283,WF%
(1,I):POKE 54290,WF%(2,I)
14180 FOR TT=1 TO 80:NEXT TT,I
14190 FOR TT=1 TO 200:NEXT:POKE 54296,0
14200 POKE 54276,1:POKE 54283,1:POKE 542
90,1
14210 RETURN
```

Mit diesem Modul werden die in Modul 4 festgelegten Stimmeigenschaften mit Hilfe von POKE in den SID gesetzt. Danach werden sie nacheinander mit POKE zusammen mit der gewünschten Wellenform in die Hoch- und Niederfrequenzen gesetzt, um die Noten zu erzeugen, aus denen die Melodie besteht.

Kommentar

Zeile 14030: Der SID-Chip wird initialisiert, indem Null mit POKE in jede der Chip-Adressen gesetzt wird.

Zeilen 14040-14090: Die in Modul 4 angegebenen Stimmeinstellungen werden mit POKE in SID gesetzt.

Zeilen 14100-14130: Die Filtereinstellungen werden von jeder Stimme gemeinsam benutzt. Sie werden nur einmal mit POKE in den Chip gesetzt.

Zeilen 14140-14180: Bis zur Melodielänge (TL) werden für jede Note die Hoch- und Niederfrequenzwerte in die beiden ersten Bytes jeder Stimmadresse des SID gesetzt, gefolgt von der Wellenform, die in die fünfte Adresse jeder Stimme gesetzt wird. Dadurch wird die gewünschte Note aktiviert, die während der Dauer der Schleife in Adresse 14180 gespielt wird.

Zeile 14190: Am Ende der Melodie wird eine etwas längere Schleife benutzt, damit der Ton ausklingen kann. Danach werden die drei Wellenformen zum Schweigen gebracht.

Testen von Modul 7.1.6

Dieser Test ist recht einfach. Wurden Melodiedaten eingegeben, wurde das Programm initialisiert, mindestens eine Stimme gesetzt und die Melodie dann kompiliert, so sollte der Benutzer nun seine erste Schöpfung spielen können. Die Musterdaten in Modul 8 spielen eine Skala von C mit den beiden ersten Stimmen.

MODUL 7.1.7

```
17000 REM#******************************
17010 REM LADEN/SPEICHERN
17020 REM*******************************
17030 INPUT "■[SCHWARZ]KASSETTE EINLEGEN
, DANN [REVERS EIN]RETURN[REVERS AUS]--"
;Q$
17040 PRINT "■[GRUEN]1)SPEICHERN":PRINT"
```

```
2)LADEN"
17050 INPUT "[BLAU]EINGABE:";Q
17060 ON Q GOTO 17070,17130:RETURN
17070 OPEN 1,1,2,"MUSIC":PRINT#1,TL
17080 FOR I=0 TO 1:FOR J=0 TO 35:PRINT#1
,NO%(I,J):NEXT J,I
17090 FOR I=0 TO 2:FOR J=0 TO 6:PRINT#1,
VO%(I,J):NEXT J,I
17100 FOR I=0 TO 3:PRINT#1,FI%(I):NEXT
17110 FOR I=0 TO 2:FORJ=0TOTL:PRINT#1,LF
%(I,J);R$;HF%(I,J);R$;WF%(I,J):NEXTJ,I
17120 CLOSE1:RETURN
17130 OPEN 1,1,0,"MUSIC":INPUT#1,TL
17140 FOR I=0 TO 1:FOR J=0 TO 35:INPUT#1
,NO%(I,J):NEXT J,I
17150 FOR I=0 TO 2:FOR J=0 TO 6:INPUT#1,
VO%(I,J):NEXT J,I
17160 FOR I=0 TO 3:INPUT#1,FI%(I):NEXT
17170 FOR I=0 TO 2:FORJ=0TOTL:INPUT#1,LF
%(I,J),HF%(I,J),WF%(I,J):NEXTJ,I
17180 CLOSE1:RETURN
```

Hier handelt es sich um ein Standardmodul einer Datendatei, mit dem Melodiedaten auf Band gespeichert werden können.

MODUL 7.1.8

```
19000 REM#****************************
19010 REM DATEN FUER STIMME 1
19020 REM*****************************
19030 DATA 34,2,34,2,39,2,41,2,43,2,39,1
,38,1,36,2,48,1,48,1,0,0
20000 REM#****************************
20010 REM DATEN FUER STIMME 2
20020 REM*****************************
20030 DATA 36,2,38,2,40,2,41,2,43,2,45,2
,47,2,48,2,0,0
21000 REM#****************************
21010 REM DATEN FUER STIMME 3
21020 REM*****************************
21030 DATA 0,0
```

Kommentar

Zeilen 1900-21030: Musterdaten, mit denen eine Skala von C gespielt werden kann. Soll eine Stimme benutzt werden, so muß dennoch 0,0 für sie eingegeben werden.

Zusammenfassung

Dieses Programm ist nur der Anfang aller Möglichkeiten der Tonschöpfung mit dem Commodore 64. Das Programm ist ein Arbeitspferd, mit dem der Benutzer seine eigene Musik entwickeln kann, die dann in andere Programme übertragen werden kann, wobei nur die Felder, in denen die Musik gespeichert ist, und Modul 6 benutzt werden, das die Melodie tatsächlich spielt.

Ein Schritt weiter

Vorausgesetzt, es sind nicht zu viele Einschränkungen im Bereich des Speicherplatzes vorhanden, so kann dieses Programm auf verschiedene Art und Weise erweitert werden:

1. Warum sollte man nicht versuchen, die Wellenform für jede Note einzugeben, anstatt nur für jede Stimme Hierzu muß nur ein dritter Wert für jede zu spielende Note aufgenommen werden.
2. Sollen längere Melodien gespielt werden, warum das Programm nicht so anpassen, daß es eine Schleife mit variabler Länge benutzt, mit der die Länge der Note festgelegt wird. Auf diese Weise wird sehr viel Platz im Feld gespart. Im endgültigen Feld müßten nur noch der Notenwert und seine Länge gespeichert werden, wobei mit letzterer die Dauer der Zeitschleife festgelegt wird. Bedauerlicherweise kann dies nur für eine Stimme erfolgen, da die Zeitschleife die Länge der Note für alle drei Stimmen festgelegt.
3. Werden längere Töne mit mehr als einer Stimme benötigt, warum dann dem Programm nicht die Möglichkeit geben, Teile einer Melodie zu kompilieren, die später von einem Spielprogramm aufgenommen werden können, das die speicherplatzaufwendigen DATA-Anweisungen nicht benötigt.

Das Buch 'Der Commodore 64 in der Praxis' beruht auf einer Sammlung von soliden, hochentwickelten Programmen in Bereichen, wie Datenspeicherung, Finanzverwaltung, Grafik, Haushaltsverwaltung, Ausbildung und Geschicklichkeitsspielen. Die Programme wurden so erstellt, daß die Funktionen des C64 maximal ausgenutzt werden.

Einige der höher entwickelten Programme umfassen ein Textverarbeitungssystem und einen Texteditor, ein Musik- und ein Synthesizerprogramm, einen Sprite-Editor und ein Programm, mit dem der Benutzer in den hochauflösenden Grafikmodus gehen kann. Dies ist bei dem Basic-Standardprogramm nicht möglich.

Jedes der Programme wird im einzelnen Zeile für Zeile erläutert. Jedes der Programme besteht aus Universal-Subroutinen und Moduln, die — nachdem ihre Funktion einmal verstanden wurde — die Grundlage für andere Programme bilden, die später geschrieben werden müssen. Die detaillierte Besprechung jeder Subroutine führt zu besseren Programmier-Fähigkeiten. Außerdem erhält der Benutzer durch diese Sammlung eine Vielzahl von praktischen Anwendungsprogrammen, die es sonst nur als teuere Software zu kaufen gibt.

Der Autor, David Lawrence, ist Autor verschiedener Bücher über Homecomputer und schreibt regelmäßig Beiträge für 'Popular Computing Weekly'.



Commodore

Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Kinskygasse 40–44
A-1232 Wien

Artikel-Nr. 556435/4.85 Änderungen vorbehalten ISBN 3-89133-011-1